МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Одеська національна академія зв'язку ім. О. С. Попова

Кафедра інформаційних технологій

ОСНОВИ ПРОГРАМУВАННЯ

Частина 2

ОПРАЦЮВАННЯ СТРУКТУРОВАНИХ ТИПІВ

Методичні вказівки до лабораторних і практичних робіт для студентів напряму 6.050103 – Програмна інженерія

Одеса 2014

УДК 004.43 План НМВ 2014 р.

Укладачі: **Трофименко О.Г., Прокоп Ю.В., Швайко І.Г., Буката Л.М.**

Рецензент: к.т.н., доцент Флейта Ю.В.

Методичні вказівки призначені для студентів при підготовці до лабораторних і практичних занять з дисципліни "Основи програмування". Оскільки ця дисципліна передбачає значну кількість лабораторних і практичних занять, методичний матеріал для їх підготовки було розбито на декілька частин.

У другій частині розглянуто засоби Visual C++ з опрацювання структурованих типів даних (масивів, рядків, структур), робота з вказівниками, засоби динамічного керування пам’яттю тощо. Саме цим темам, згідно з навчальною програмою, присвячені з десятої по двадцяту лабораторні роботи курсу. Кожна із наведених робіт містить короткі теоретичні відомості, приклади створювання програмних проектів засобами Visual C++, питання для самоконтролю та варіанти індивідуальних завдань різних рівнів складності для виконання їх на комп’ютері.

Методичні вказівки будуть корисними студентам спеціальності "Програмна інженерія", які вивчають дисципліну "Основи програмування" для набуття навиків програмування з метою подальшого застосовування цих навиків у власній повсякденній і майбутній професійній діяльності; також стануть у нагоді користувачам персональних комп’ютерів, які бажають навчитися програмувати в середовищі Visual C++.

СХВАЛЕНО
на засіданні кафедри
інформаційних технологій
і рекомендовано до друку.

Протокол № 2 від 01.10.2014 р.

ЗАТВЕРДЖЕНО
методичною радою академії зв’язку.

Протокол № від . .2014 р.

Лабораторна робота № 10

Одновимірні масиви

Мета роботи: набути практичних навиків програмного опрацювання одновимірних масивів.

Теоретичні відомості

Масив у програмуванні – це упорядкована сукупність однотипних елементів.

При оголошенні масивів у квадратних дужках зазначається кількість елементів, а нумерація елементів завжди розпочинається з нуля.

Одновимірний масив оголошується в програмі таким чином:

<тип_даних> *<ім'я_масиву>* [*<розмір_масиву>*];

Приклади оголошення масивів:

```
int a[125];    double vector[100];
```

Кожен елемент масиву однозначно можна визначити за ім'ям масиву й індексами. *Індекси* визначають місцезнаходження елемента в масиві і записують після імені масиву в квадратних дужках, тобто використовуючи ім'я масиву та індекс, можна звертатися до елементів масиву:

<ім'я_масиву> [*<значення_ індексу>*]

Значення індексів повинні бути в діапазоні від нуля до величини, на одиницю меншу, ніж розмір масиву, визначений при його оголошенні, оскільки в С++ нумерація індексів розпочинається з нуля.

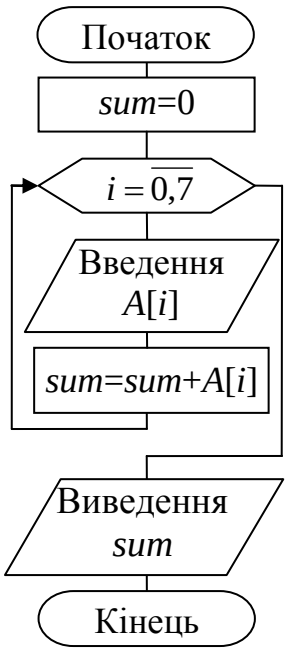
Наприклад, команда

```
int A[10];
```

оголошує масив з ім'ям `A`, який містить 10 цілих чисел: `A[0]` – перший елемент, `A[1]` – другий, `A[9]` – останній.

Приклади програм

Приклад 1. Ввести масив з 8 дійсних чисел і обчислити суму елементів масиву.



Текст програми та блок-схема:

```
#include <iostream>
using namespace std;
int main()
{ setlocale(0,".1251");
  double A[8], sum = 0;
  cout << "Введіть 8 чисел" << endl;
  for (int i=0; i<8; i++)
  { cin >> A[i];     //  введення елементів масиву
    sum += A[i];     //  обчислення суми елементів
  }
  cout << "Сума елементів масиву дорівнює "<< sum << endl;
```

```
  system ("pause>>void");
  return 0;
}
```

Результати роботи:

Елементи масиву можна водити через пробіл:

```
Введіть 8 чисел:
1   2  3  4  5  6  7  8
Сума елементів масиву = 36
```

Числа також можна ввести через *Enter*:

```
Введіть 8 чисел:
0.1
-72
3.5
12.7
50
4.6
2
0.2
Сума елементів масиву = 1.1
```

Приклад 2. Ввести масив з 8-ми дійсних чисел і видалити з нього всі від'ємні елементи.

Текст програми:

```
#include <iostream>
using namespace std;
int main()
{ setlocale(0,".1251");
 double a[8]; int i,j,n=8;
 cout << "Введіть масив з 8-ми дійсних чисел:\n ";
 for (i=0; i<n; i++) cin>>a[i];        //  введення масиву
 for (i=0; i<n; i++)
   if (a[i]<0)
   { for (j=i; j<n; j++) a[j]=a[j+1];  // видалення елемента з індексом j
     n--;    i--;
   }
  cout<<"\nУ перетвореному масиві "<<n<<" елементи(ів): \n ";
  for(i=0; i<n; i++) cout<<a[i]<<"\t"; // виведення масиву
  cin.get();  cin.get();
  return 0;
 }
```

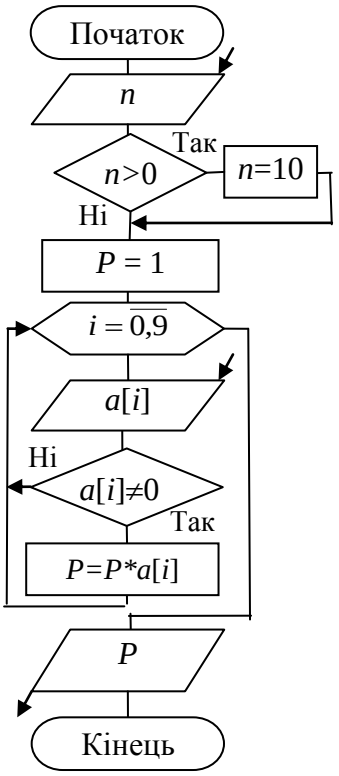
Результати роботи:

```
Введіть масив з 8-ми дійсних чисел:
 -4   3.5  0  -2.3  39  -10  45  9

У перетвореному масиві 5 елементи(ів):
 3.5   0   39   45   9
```

Приклад 3. Ввести за допомогою форми масив до 10-ти цілих чисел та обчислити добуток ненульових елементів масиву.

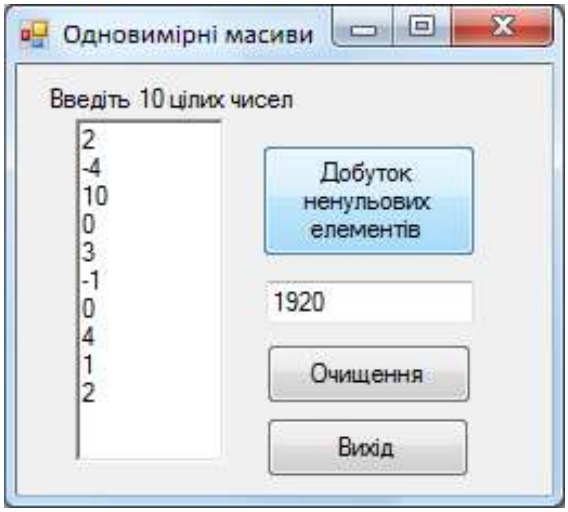
Розв'язок. Вводити масиви можна у компонент `richTextBox`.

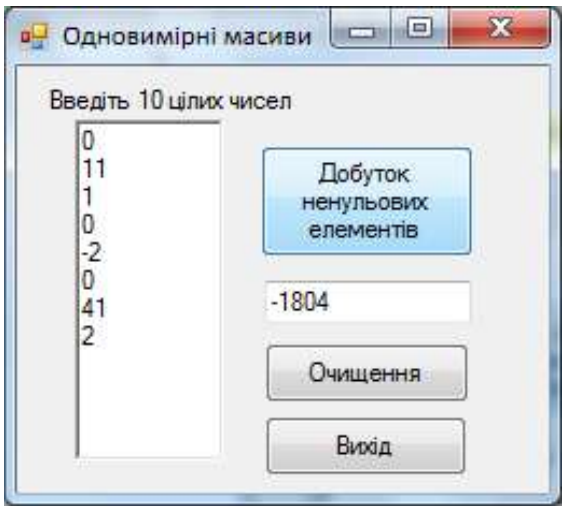
Текст програми та блок-схема:




```
private: System::Void button1_Click(System::Object^
                              sender, System::EventArgs^ e)
{
  int A[10], p=1, n=richTextBox1->Lines->Length;
  if (n>10) n=10;
  for (int i=0; i<n; i++)
  { A[i]=Convert::ToInt32(richTextBox1->Lines[i]);
    if (A[i]!=0) p*=A[i];
  }
   textBox1->Text=Convert::ToString(p);
}

private: System::Void button2_Click(System::Object^
                              sender, System::EventArgs^ e)
{
  richTextBox1->Clear(); textBox1->Clear();
}
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)
{
  Close();
}
```

Результати роботи:





Приклад 4. Ввести масив із 9-ти дійсних чисел, обчислити найменший елемент масиву та його індекс.

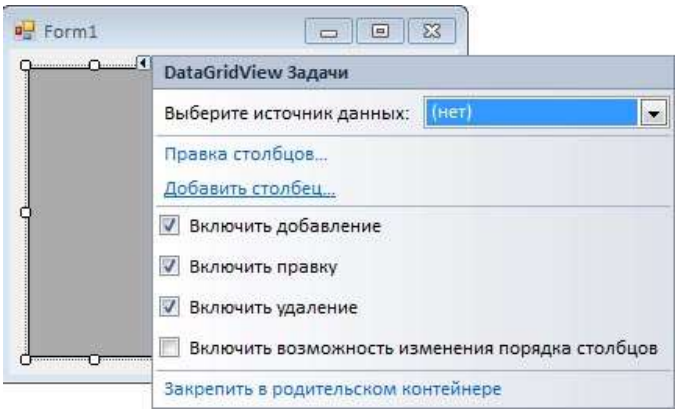
Розв'язок. У вигляді словесного алгоритму пошук мінімального елемента масиву і його індексу можна записати в такий спосіб:

1) вважаємо перше число за найменше. Для цього присвоюємо перший елемент масиву `A[0]` у змінну `min`, а змінній `ind` – значення 0 (тобто індекс першого елемента масиву);

2) у циклі переглядаємо всі елементи. Якщо зустрічаємо число, яке є менше за `min`, запам'ятовуємо це число у `min`, а його індекс – в `ind`.

Коли цикл завершиться, у змінній `min` буде зберігатися найменший елемент масиву, а його індекс – у змінній `ind`. Оскільки нумерація індексів масиву розпочинається з 0, а не з 1, як у повсякденному житті, виветься значення індексу, збільшене на 1.

Крім компонента `richTextBox`, масиви також можна вводити (чи виводити) в компонент `dataGridView`. При розміщенні його на формі автоматично сформується діалогове вікно *DataGridView Задачи*, в якому слід вибрати команду *Добавить столбец.* У вікні, що відкриється, слід натиснути кнопку *Добавить*, при цьому сформується один стовпець. (Кожному стовпцю можна задавати власне ім'я та вибирати тип даних. За замовчуванням перший створений стовпець матиме ім'я `Column1` і тип даних, який надає можливість введення-виведення тексту.)



Оскільки для введення одновимірного масиву достатньо одного стовпця, більше стовпців створювати не треба, а просто натиснути кнопку *Закрыть*.

Далі слід вибрати значення `False` для трьох властивостей `dataGridView`:

– `ColumnHeadersVisible` (заголовок стовпця);

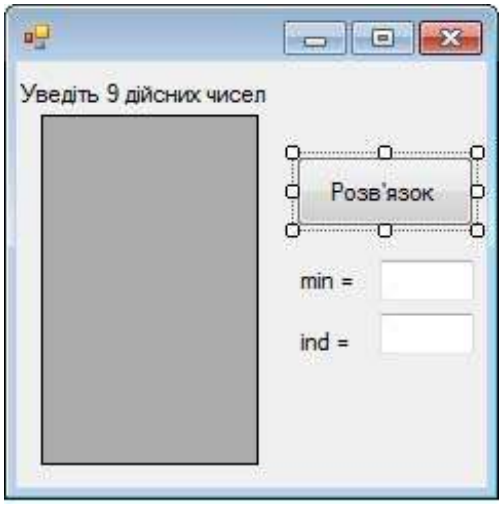
– `RowHeadersVisible` (стовпець із заголовками рядків);

– `AllowUserToAddRows` (дозвіл на автоматичне долучення рядків).

Крім вже налаштованого елемента `dataGridView` слід на панелі елементів[1] вибрати і розмістити на формі дві командні кнопки `button`, два `textBox` для виведення результатів та три `label` для створення пояснювальних підписів.

[1] Якщо в даний момент Ви не бачите панель елементів, її можна показати командою меню *Вид / Панель элементов* (*View/Toolbox*) а бо клавішами *Ctrl+Alt+x*.

Після остаточних налаштувань форма набуде вигляду →

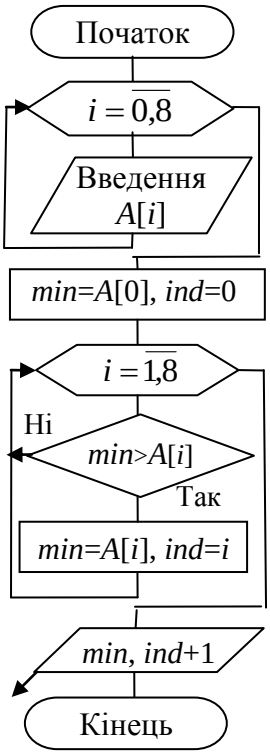


Перед написанням програмного коду слід зазначити ще одну особливість `dataGridView`: рядки можна долучати лише у програмі. Саме тому слід подвійним клацанням миші по формі створити функцію `Form1_Load` та вписати в неї команду створення 9-ти рядків:

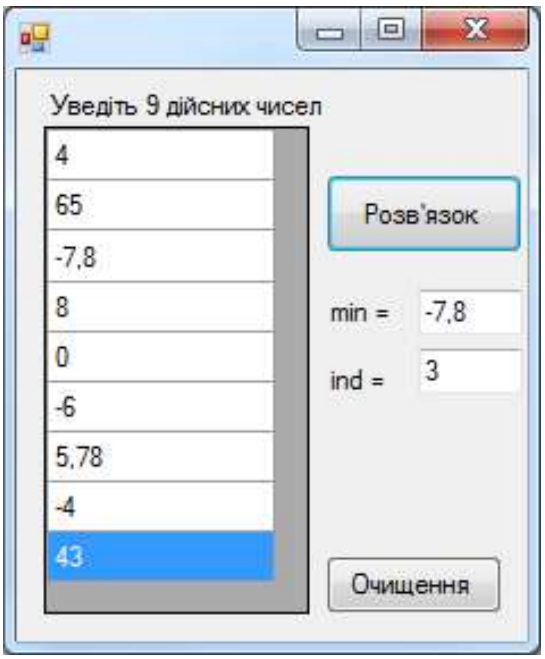
```
dataGridView1->Rows->Add(9);
```

Текст програми та блок-схема:

```
private: System::Void Form1_Load
                        (System::Object^ sender, System::EventArgs^ e)
{  dataGridView1->Rows->Add(9);
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{ double a[9], min;  int i, ind;
 for (i=0; i<9; i++)
  a[i]=Convert::ToDouble(dataGridView1->Rows[i]->Cells[0]->Value);
 min=a[0]; ind=0;
 for (i=1; i<9; i++)
   if (a[i]<min) { min=a[i]; ind=i;}
 textBox1->Text = min.ToString();
 textBox2->Text = (ind+1).ToString();
}
private: System::Void button2_Click(System::Object^ sender,
                                    System::EventArgs^ e)
{ for (int i=0; i<9; i++)
    dataGridView1->Rows[i]->Cells[0]->Value = "";
  textBox1->Clear();  textBox2->Clear();
}
```



Результати роботи:



Приклад 5. Ввести масив із 12-ти дійсних чисел і впорядкувати (відсортувати) його за зростанням значень.

Розв'язок. У наведеній нижче програмі запропоновано найбільш поширений метод "бульбашки" (bublesort). Алгоритм сортування полягає у переставлянні двох сусідніх порівнюваних елементів масиву, якщо вони йдуть у неправильному порядку.

Оскільки одноразового проходження по елементам масиву недостатньо для розв'язання завдання, в програмі організовано два цикли: зовнішній цикл (по `i`) дозволяє перебрати всі елементи масиву для порівняння з іншими елементами, доступ до яких організовано у вкладеному циклі (по `j`). При цьому останнє значення параметра циклу `i` у циклі `for` має бути на один менше за кількість усіх елементів, щоб на останній ітерації можна було порівнювати передостанній елемент з останнім.

Для переставляння двох елементів застосовується тимчасова змінна `temp`:

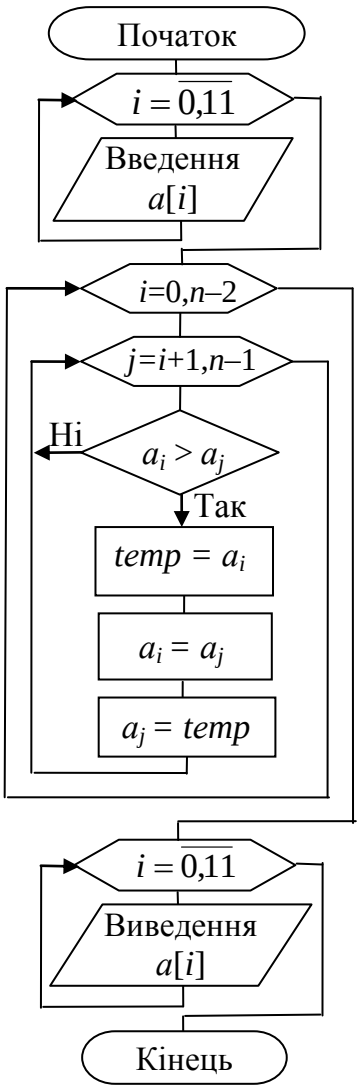
```
temp = a[i];   // Запам'ятати значення a[i],
a[i] = a[j];   // в a[i] записати a[j],
a[j] = temp;   // а в a[j] записати те, що спочатку було в a[i]
```

Щоб краще зрозуміти необхідність тимчасової змінної, слід уявити собі склянку соку і чашку кави і спробувати перелити сік у чашку, а каву – у склянку. Це неможливо зробити без додаткового посуду, до якого спочатку треба перелити вміст, наприклад, склянки.

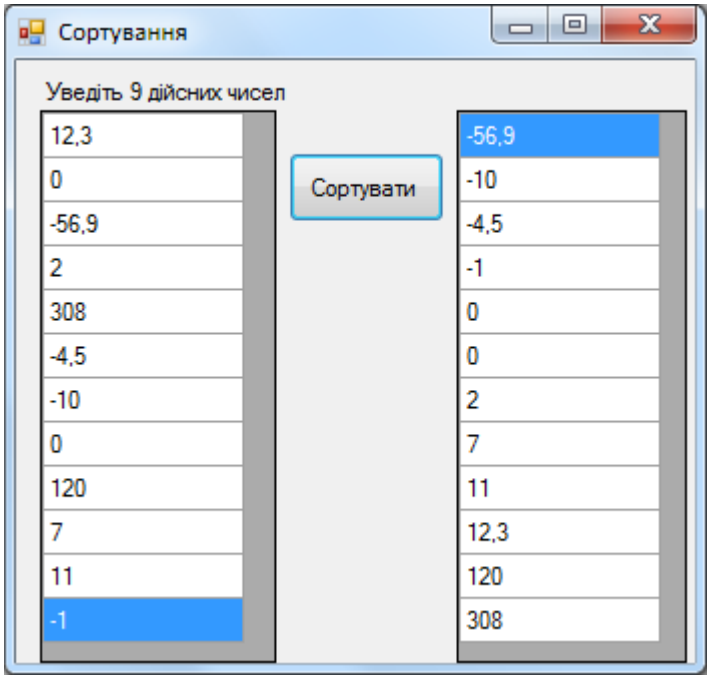
Якщо в умові порівняння елементів записати знак ">", то елементи сортуватимуться за зростанням, а якщо знак "<" – за спаданням.

Текст програми та блок-схема:

```
private: System::Void Form1_Load(System::Object^ sender,
                                 System::EventArgs^ e)
{ dataGridView1->Rows->Add(12);
  dataGridView2->Rows->Add(12);
}
private: System::Void button1_Click(System::Object^
                                sender, System::EventArgs^ e)
{ double a[12];
  for (int i=0; i<12; i++)
  a[i]=Convert::ToDouble(dataGridView1->Rows[i]->
                                        Cells[0]->Value);
 for (int i=0; i<11; i++)
  for (int j=i+1; j<12; j++)
   if (a[i]>a[j])
   { double temp=a[i];
     a[i]=a[j];
     a[j]=temp;
   }
```



```
 for (int i=0; i<12; i++)
  dataGridView2->Rows[i]->Cells[0]->Value = a[i].ToString();
}
```



Питання для самоконтролю

1) Дайте визначення масиву в програмуванні?
2) Запишіть оголошення одновимірного масиву з 25-ти дійсних чисел.
3) Наведіть оператор, який присвоює першому елементу масиву A з 25-ти дійсних чисел нульове значення.
4) Виберіть ім'я змінної, значенням якої є сума елементів масиву, в наведених трьох фрагментах програм:

а)
```
s1=0;
for(i=0;i<10;i++)
  s1 += a[i];
```

б)
```
s2=1;
for(i=0;i<10;i++)
  s2 *= a[i];
```

в)
```
s3=0;
for(i=0;i<10;i++)
  if(a[i]>0) s3++;
```

5) Запишіть ім'я змінної, значенням якої є кількість додатних елементів масиву, в наведених трьох фрагментах програм:

а)
```
S1=0;
for(i=0;i<10;i++)
 S1 += a[i];
```

б)
```
S2=1;
for(i=0;i<10;i++)
 S2 *= a[i];
```

в)
```
S3=0;
for(i=0;i<10;i++)
  if(a[i]>0) S3++;
```

6) Запишіть ім'я змінної, значенням якої є максимальний з елементів масиву, в наведених трьох фрагментах програм:

а)
```
S1=0;
for(i=0;i<10;i++)
 S1 += a[i];
```

б)
```
S2=a[0];
for(i=1;i<10;i++)
  if(a[i]>S2)S2=a[i];
```

в)
```
S3=0;
for(i=0;i<10;i++)
  if(a[i]>0)S3=S3+1;
```

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи скласти схему алгоритму і написати програму мовою С++ для розв'язання індивідуальних завдань з опрацювання одновимірних масивів (завдання вибрати з табл. 10.1 … 10.3).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 10.1

Індивідуальні завдання середнього рівня складності

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	15	цілий	Обчислити кількість та суму парних елементів масиву
2	10	цілий	Обчислити середнє арифметичне додатних елементів
3	8	цілий	Обчислити факторіал значення останнього елемента
4	12	дійсний	Обчислити добуток елементів, значення яких менше 6-ти
5	14	цілий	Обчислити середнє арифметичне непарних елементів
6	18	дійсний	Впорядкувати за спаданням першу половину масиву
7	11	цілий	Розмістити елементи масиву у зворотному порядку
8	14	дійсний	Розмістити елементи масиву у порядку зростання значень їх модулів
9	16	цілий	Обчислити суму елементів масиву, кратних 3
10	14	дійсний	Обчислити суму елементів, абсолютне значення яких не перевищує 10
11	17	цілий	Обчислити середнє арифметичне мінімального та максимального елементів масиву
12	9	дійсний	Вивести кількість елементів, значення яких більше за значення першого елемента масиву
13	15	цілий	Визначити індекси мінімального і максимального елементів масиву
14	10	дійсний	Обчислити суму елементів масиву, значення яких належать проміжку [3, 6]
15	8	цілий	Обчислити добуток непарних елементів масиву
16	12	дійсний	Визначити номери мінімального і максимального елементів масиву
17	20	цілий	Впорядкувати за зростанням другу половину масиву
18	18	цілий	Обчислити середнє арифметичне елементів, значення яких більше значення останнього елемента масиву
19	11	дійсний	Обчислити кількість додатних елементів, значення яких менше 20-ти
20	9	дійсний	Обчислити модуль суми всіх від'ємних елементів, суму всіх додатних і різницю між значеннями цих сум
21	16	цілий	Обчислити середнє арифметичне елементів, кратних 5
22	19	дійсний	Визначити мінімальний із додатних елементів

Закінчення табл. 10.1

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
23	17	цілий	Обчислити кількість додатних, від'ємних і нульових елементів
24	8	дійсний	Обчислити добуток одноцифрових елементів масиву
25	7	дійсний	Обчислити суму тільки двоцифрових елементів
26	18	дійсний	Визначити максимальний і мінімальний елементи, наскільки максимальний елемент є більшим за мінімальний
27	11	цілий	Обчислити процентний вміст додатних, від'ємних і нульових елементів
28	10	дійсний	Обчислити суму квадратів тих чисел, модуль яких більше значення 2.5
29	12	дійсний	Визначити найбільший з парних додатних елементів
30	9	дійсний	Обчислити суму модулів усіх від'ємних елементів масиву

Таблиця 10.2

Індивідуальні завдання базового рівня складності

№ вар	Розмір масиву	Тип даних	Індивідуальне завдання
1	15	цілий	Поміняти місцями мінімальний елемент з передостаннім
2	10	дійсний	Обчислити суму додатних непарних елементів і замінити парні елементи масиву на цю суму
3	12	цілий	Перевірити, чи є масив упорядкованим за зростанням
4	8	дійсний	Обчислити факторіал першого елемента масиву, значення якого менше 8-ми
5	14	цілий	Поміняти місцями максимальний елемент з першим
6	18	дійсний	Обчислити новий масив як різницю елементів вихідного масиву та їх середнього арифметичного
7	11	цілий	Замінити всі від'ємні елементи мінімальним
8	14	дійсний	Обчислити кількість елементів, що перевищують середнє арифметичне всіх елементів
9	7	цілий	Створити новий масив, розмістити спочатку всі додатні елементи і нулі, після чого – від'ємні, зберігаючи порядок їх слідування
10	19	дійсний	Поміняти місцями максимальний елемент з мінімальним
11	17	цілий	Обчислити факторіал індексу максимального елемента
12	9	дійсний	Замінити мінімальний і максимальний елементи значенням середнього арифметичного всіх елементів
13	15	цілий	Поміняти місцями першу половину масиву з другою
14	10	цілий	Замінити парні по значенню (не по індексу) елементи на 0
15	8	дійсний	Поміняти місцями елементи, які стоять у масиві поряд: 1 і 2, 3 і 4 і т. д.

Закінчення табл. 10.2

№ вар	Розмір масиву	Тип даних	Індивідуальне завдання
16	12	дійсний	Видалити з масиву перші три елементи масиву
17	20	цілий	Визначити найменший серед парних додатних елементів
18	18	дійсний	Видалити з масиву всі елементи від початку і до найбільшого елемента
19	11	цілий	Видалити з масиву парні за значенням елементи масиву
20	9	дійсний	Створити новий масив із залишків від ділення націло елементів масиву на 3
21	16	цілий	Визначити найменший серед від'ємних елементів масиву
22	19	дійсний	Замінити всі нульові елементи значенням мінімального елемента
23	17	цілий	Обчислити суму від'ємних елементів, розміщених після максимального елемента масиву
24	8	дійсний	Замінити всі від'ємні елементи на нулі, а додатні – на одиниці
25	7	цілий	Замінити всі непарні елементи масиву одиницями
26	18	цілий	Обчислити кількість елементів масиву, розміщених після першого нульового елемента
27	11	цілий	Видалити з масиву нульові елементи масиву
28	10	дійсний	Видалити з масиву найменший елемент масиву
29	16	дійсний	Видалити з масиву перший та останній елементи масиву
30	12	цілий	Обчислити суму парних елементів, розміщених після мінімального елемента масиву

Таблиця 10.3

Індивідуальні завдання високого рівня складності

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	до 9	цілий	Ввести два масиви й обчислити кількість однакових елементів у них
2	до 10	дійсний	Ввести два масиви і побудувати третій з упорядкованих за зростанням значень елементів обох масивів
3	до 12	цілий	Ввести масив, в якому кожний елемент є 0, 1 або 2, переставити елементи масиву так, щоб спочатку були розміщені всі 1, потім всі 0 і, наприкінці, всі 2
4	до 8	дійсний	Ввести масив і число С. Переставити числа в масиві так, щоб спочатку були розміщені всі елементи менші за значення С, потім – більші С, зберігаючи порядок їх розміщення
5	до 14	цілий	Визначити перше число, яке є присутнім у кожному з 3-х масивів, значення в цих масивах розміщено за зростанням
6	до 7	дійсний	Ввести два масиви та створити третій зі спільних елементів масивів

Продовження табл. 10.3

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
7	до 11	цілий	Ввести масив, в якому тільки два однакові елементи. Визначити їхні індекси
8	до 14	дійсний	Ввести масив і число L. Створити новий масив з елементів, менших за L, і впорядкувати новий масив за спаданням
9	до 19	цілий	Визначити пару сусідніх елементів масиву, значення яких є найближчими один до одного, тобто значення $\lvert x_{i+1} - x_i \rvert$ є мінімальним
10	до 12	дійсний	Обчислити добуток P від'ємних елементів з парними індексами, які не перевищують введеного числа L, і поділити всі додатні елементи на P
11	до 17	цілий	Ввести масив і створити на його основі два нових масиви: перший з елементів з непарними індексами, другий – з елементів, кратних 5
12	до 9	дійсний	Ввести масив і число P. Визначити елемент масиву, значення якого найближче до P, тобто значення $\lvert x_i - P \rvert$ є мінімальним
13	до 15	цілий	Впорядкувати масив так, щоб всі додатні числа були розміщені спочатку за зростанням, а всі від'ємні – наприкінці за спаданням
14	до 10	дійсний	Ввести два масиви і визначити кількість неоднакових елементів у них
15	до 8	цілий	Ввести два масиви та замінити нулями ті елементи першого масиву, яких немає у другому
16	до 12	дійсний	Обчислити суму S від'ємних елементів, які не перевищують заданого числа L, і поділити останній додатний елемент на S
17	до 20	цілий	Ввести масив і число K. Створити два нових масиви: перший – з елементів, менших за значення K, другий – з елементів, більших за K, зберігаючи порядок їх розміщення
18	до 18	дійсний	Ввести масив, який містить багато нульових елементів. Замінити всі групи підряд розміщених нулів на значення кількості нулів
19	до 11	цілий	Обчислити суму S додатних непарних елементів і замінити всі парні елементи масиву на S
20	до 9	дійсний	Ввести два масиви, визначити максимальні елементи кожного з них і поміняти їх місцями
21	до 16	цілий	Ввести масив і на його основі створити новий з від'ємних і нульових елементів з парними індексами
22	до 19	дійсний	Ввести два масиви, визначити мінімальний елемент першого і максимальний елемент другого масиву і поміняти їх місцями

Закінчення табл. 10.3

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
23	до 17	цілий	Ввести масив і на його основі створити два нових масиви: перший – з парних елементів, другий – з елементів, кратних 3
24	до 8	дійсний	Ввести два масиви і замінити нулями ті елементи другого масиву, які є в першому
25	до 7	цілий	Ввести два масиви і поміняти місцями максимальний елемент першого масиву з першим елементом другого, який більше 5-ти
26	до 18	дійсний	Ввести масив і на його основі створити новий з додатних елементів, більших за середнє арифметичне всіх елементів
27	до 15	цілий	Ввести масив і число L. Створити новий масив з елементів, більших за L, і впорядкувати створений масив за зростанням
28	до 10	дійсний	Визначити пару сусідніх елементів масиву, значення яких максимально різняться один від одного, тобто значення $\lvert x_{i+1} - x_i \rvert$ є максимальним
29	до 16	цілий	Ввести масив, який містить багато нульових елементів. Замінити всі групи підряд розміщених нулів на один нуль
30	до 12	дійсний	Ввести два масиви, визначити мінімальні елементи кожного з них і поміняти їх місцями

Лабораторна робота № 11

Опрацювання одновимірних масивів у функціях

Мета роботи: набути практичних навиків програмного опрацювання одновимірних масивів у функціях засобами Visual C++.

Теоретичні відомості

При опрацюванні масивів у функціях передавання їх у якості аргументів завжди здійснюється за адресою, тобто передається адреса першого елемента (початок масиву), а доступ до кожного з елементів масиву здійснюється як певний зсув (обчислюваний через індекси) від початку масиву.

Передавання одновимірних масивів до функцій у якості аргументів можна організувати одним з трьох способів:

```
double fun (int a[10]);
double fun (int a[]);
double fun (int *a);
```

За першим способом явно зазначено кількість елементів масиву. У другій синтаксичній формі константний вираз у квадратних дужках є відсутній. За третього способу передається вказівник як посилання на масив, явно визначений в основній програмі. Оскільки для другого і третього способів інформація про кількість елементів у прототипі відсутня, то такі форми припустимі, коли кількість елементів масиву є глобальними змінними чи константами. Але доцільніше передавати розмірність одновимірного масиву до функції окремим параметром, оскільки це зробить таку функцію більш універсальною, адже вона зможе коректно опрацьовувати масиви з різною кількістю елементів.

```
double fun (int a[], int n);
double fun (int *a, int n);
```

Хоча наведені записи мають різний синтаксис, насправді вони є рівнозначними, оскільки авторами стандарту С для більшої ясності було вирішено, що масив оголошений як параметр функції є **вказівником**. При цьому навіть явно зазначене число у квадратних дужках ігнорується.

Якщо функція для опрацювання елементів масиву не повинна передавати результат як одне значення, а лише переставляє елементи масиву місцями або змінює їхні значення, то таку функцію оголошують з типом результату `void` (нема величини, що повертається). Оскільки сам масив передається до функції за посиланням, то будь-які змінювання значень елементів масиву у функції буде видно і в основній програмі, яка викликає цю функцію (див. приклад програми 3).

Приклади програм

Приклад 1. Створити функцію обчислення середнього арифметичного значення ненульових елементів цілочислового масиву та за допомогою цієї функції визначити середнє арифметичне ненульових елементів для введеного масиву з 10-ти цілих чисел: спочатку для всіх елементів, а тоді для першої половини масиву.

Схеми алгоритму функції та основної програми:

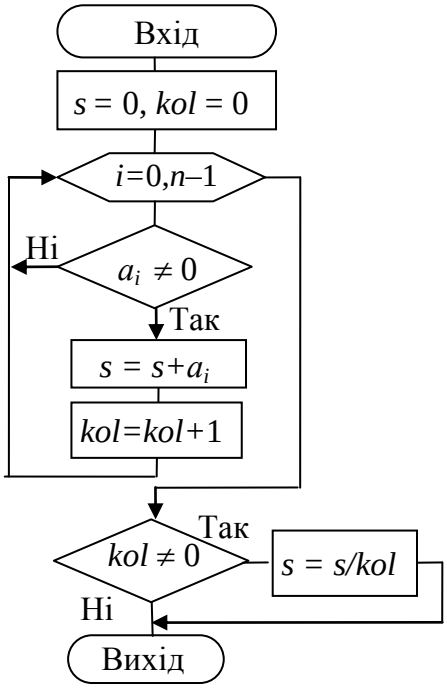


Схема алгоритму функції **sr()**

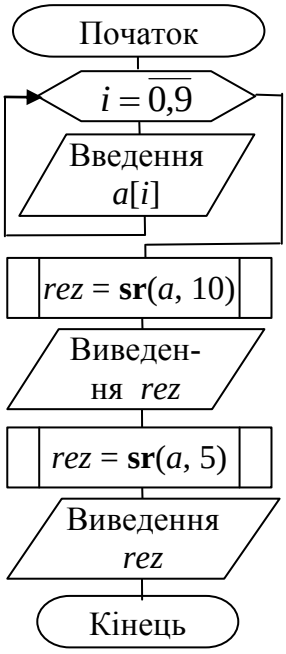


Схема алгоритму до основної програми

Текст програмного коду та схема алгоритму:

```
#include <iostream>
using namespace std;
double sr(int a[], int n)
{ double s=0; int i, kol=0;
  for (i=0; i<n; i++)
    if (a[i]) { s+=a[i]; kol++; }
  if (kol) s/=kol;
  return s;
}
int main ()
{ setlocale(0,".1251");
  int a[10], i;
  cout<<" Введіть 10 цілих чисел:\n";
  for(i=0; i<10; i++) cin >> a[i];
  cout<<"\n Середнє арифметичне "<<10<<" елементів - " <<sr(a,10) <<endl;
  cout<<" Середнє арифметичне "<< 5 << " елементів - " << sr(a,5) <<endl;
  system ("pause>>void");
  return 0;
}
```

Результати роботи консольного додатка:

```
Введіть 10 цілих чисел:
6   -8   0   3   1   34   -5   1   0   2
Середнє арифметичне 10 елементів -  4.25
Середнє арифметичне 5 елементів -  0.5
```

Приклад 2. Створити функцію визначення максимального і мінімального елементів масиву з 10-ти дійсних чисел та організувати її виклик для введеного масиву.

Схеми алгоритму функції та основної програми:

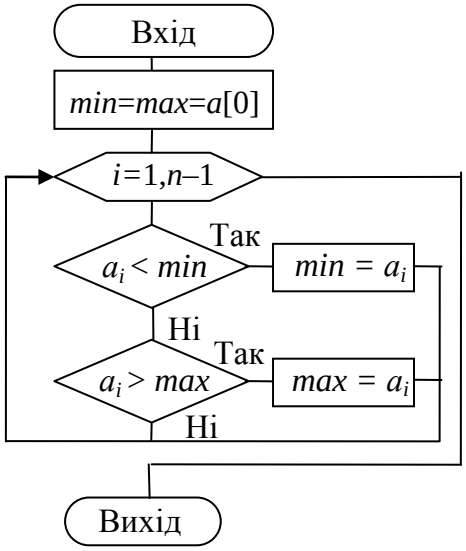


Схема алгоритму функції **MinMax()**

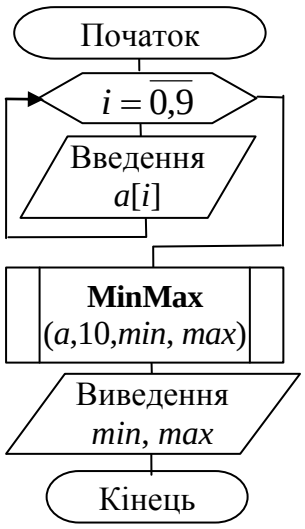


Схема алгоритму до основної програми

Текст функції та основної програми:

```
#include <iostream>
using namespace std;

void MinMax (double a[], int n, double& min, double& max)
{ min=max=a[0];
  for (int i=1; i<n; i++)
  { if (a[i]<min) min=a[i];
    if (a[i]>max) max=a[i];
  }
}
int main ()
{ setlocale(0,".1251");
  double a[10], min, max;
  cout<<"Введіть 10 чисел:\n";
  for (int i=0; i<10; i++) cin >> a[i];
  MinMax(a, 10, min, max);
  cout<<" Мінімальне значення: "<< min << endl;
  cout<<" Максимальне значення: "<< max << endl;
  system ("pause>>void");
  return 0;
}
```

Результати роботи консольного додатка:

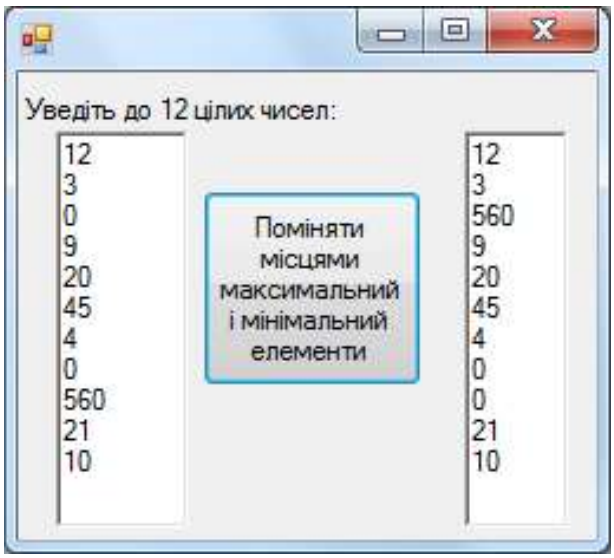
```
Введіть 10 чисел:
1.2   0   -3.8   12   3   34.5   7   0   -54   14

Мінімальне значення:  -54
Максимальне значення:  34.5
```

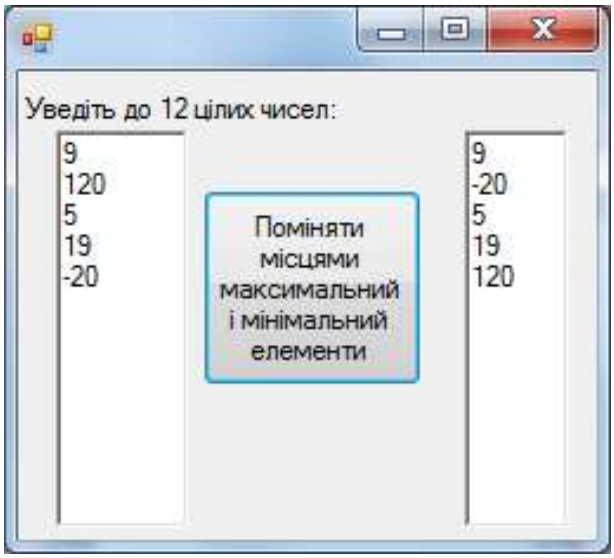
Приклад 3. Створити функцію для обміну місцями максимального та мінімального елементів масиву і перевірити правильність роботи функції для масиву розмірністю до 12-ти цілих чисел.

Текст функції та основної програми:

```
void change(int a[], int n)
{
  int imin=0, imax=0, tmp;
  for (int i=0; i<n; i++)
  {
    if(a[imin]>a[i]) imin=i;
    if(a[imax]<a[i]) imax=i;
  }
    tmp=a[imin];
    a[imin]=a[imax];
    a[imax]=tmp;
}
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{
  int a[12],i,n=richTextBox1->Lines->Length;
  if (n>12) n=12;
  for (i=0; i<n; i++) a[i]=Convert::ToInt32(richTextBox1->Lines[i]);
  change(a,n);
  richTextBox2->Clear();
  for (i=0; i<n; i++) richTextBox2->Text+=Convert::ToString(a[i])+"\n";
}
```

Результати роботи:





Питання та завдання для самоконтролю

1) Як передаються масиви до функцій у якості їх аргументів?
2) Який тип матиме функція, яка повинна сортувати масив? Обґрунтуйте відповідь.
3) Яку інформацію про функції надають такі їх заголовки:

```
int fun(double x[], int n);
void fun(double x[], int n);
double fun(double x[], int n);
double fun(double x[], int n, double& z);
```

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи скласти схеми алгоритмів функцій та основних програм, а також написати програмний код функцій та основних програм мовою С++ для розв'язання завдань, поданих у табл. 11.1 … 11.2 відповідно до індивідуального варіанта.

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 11.1

Варіанти завдань для створення функції з одним або двома результатами

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	15	цілий	Обчислити факторіал першого елемента масиву, значення якого менше 8-ми
2	10	цілий	Обчислити кількість елементів масиву, розміщених після першого нульового елемента
3	8	цілий	Обчислити факторіал значення першого додатного елемента
4	12	дійсний	Визначити найбільший серед від'ємних елементів масиву
5	14	цілий	Обчислити суму і кількість парних елементів масиву
6	18	дійсний	Обчислити суму елементів, абсолютне значення яких не перевищує 10-ти
7	11	цілий	Обчислити суму та кількість елементів масиву, кратних 3
8	14	дійсний	Обчислити факторіал індексу максимального елемента
9	16	цілий	Обчислити середнє арифметичне мінімального та максимального елементів масиву
10	14	дійсний	Обчислити кількість елементів, значення яких більше за значення першого елемента масиву
11	17	цілий	Обчислити середнє арифметичне елементів, значення яких більше за значення останнього елемента масиву
12	9	дійсний	Визначити індекси мінімального та максимального елементів масиву

Закінчення табл. 11.1

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
13	15	цілий	Обчислити модуль суми всіх від'ємних елементів, суму всіх додатних
14	10	дійсний	Обчислити суму від'ємних елементів, розміщених після максимального елемента масиву
15	8	цілий	Визначити мінімальний із додатних елементів
16	12	дійсний	Обчислити різницю між значеннями суми всіх додатних елементів і модулем суми всіх від'ємних елементів
17	20	цілий	Обчислити добуток одноцифрових елементів масиву
18	18	цілий	Обчислити суму елементів масиву, значення яких належать проміжку [3, 6]
19	11	дійсний	Визначити максимальний і мінімальний елементи, наскільки максимальний елемент є більшим за мінімальний?
20	9	дійсний	Обчислити кількість елементів, що перевищують середнє арифметичне всіх елементів
21	16	цілий	Визначити найбільший з парних додатних елементів
22	19	дійсний	Перевірити, чи є масив упорядкованим за зростанням
23	17	цілий	Обчислити суму тільки двоцифрових елементів
24	8	цілий	Обчислити середнє арифметичне елементів, кратних 5
25	7	цілий	Визначити найменший серед парних додатних елементів
26	18	дійсний	Обчислити кількість додатних елементів, значення яких менше 20-ти
27	11	дійсний	Обчислити суму модулів усіх від'ємних елементів масиву
28	10	дійсний	Обчислити суму квадратів тих чисел, модуль яких більше значення 2.5
29	12	цілий	Обчислити суму парних елементів, розміщених після мінімального елемента масиву
30	9	дійсний	Обчислити кількість додатних і від'ємних елементів

Таблиця 11.2

Варіанти завдань створення функції для змінення елементів масиву

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	15	цілий	Обчислити суму додатних непарних елементів і замінити всі парні елементи масиву на цю суму
2	10	дійсний	Поміняти місцями мінімальний елемент з передостаннім
3	12	цілий	Замінити всі від'ємні елементи на максимальний
4	8	дійсний	Поміняти місцями першу половину масиву з другою
5	14	цілий	Замінити парні за значенням (не по індексу) елементи на 0
6	11	цілий	Обчислити суму додатних непарних елементів і замінити парні елементи масиву на цю суму

Закінчення табл. 11.2

№ вар	Розмір масиву	Тип даних	Індивідуальне завдання
7	18	дійсний	Поміняти місцями елементи, які стоять у масиві поряд: 1 і 2, 3 і 4 і т. д.
8	14	цілий	Поміняти місцями максимальний елемент з першим
9	7	дійсний	Поміняти місцями два найбільші за значенням елементи
10	11	цілий	Впорядкувати за спаданням першу половину масиву
11	12	дійсний	Замінити мінімальний і максимальний елементи значенням середнього арифметичного всіх елементів
12	15	цілий	Замінити всі непарні елементи масиву одиницями
13	9	дійсний	Замінити всі нульові елементи значенням мінімального елемента
14	8	дійсний	Розмістити елементи масиву у зворотному порядку
15	12	цілий	Розмістити елементи масиву у порядку зростання значень їх модулів
16	12	дійсний	Замінити нулями ті елементи масиву, які більші за середнє арифметичне
17	16	дійсний	Впорядкувати за зростанням другу половину масиву
18	14	цілий	Замінити нулями всі елементи від початку і до найбільшого елемента
19	12	цілий	Замінити на максимальне значення масиву всі його нульові елементи
20	16	дійсний	Замінити найменший та найбільший елементи на нулі
21	9	цілий	Поміняти місцями два найменші за значенням елементи
22	16	дійсний	Замінити на значення мінімального елемента ті елементи масиву, які менші за середнє арифметичне
23	10	цілий	Замінити елементи кратні 5 на значення найбільшого непарного елемента
24	18	цілий	Замінити всі елементи кратні двом на значення найбільшого елемента
25	14	дійсний	Поміняти місцями максимальні елементи першої та другої половин масиву
26	12	цілий	Замінити парні за значенням елементи масиву на 0
27	10	цілий	Замінити всі парні елементи масиву на значення останнього елемента масиву
28	11	дійсний	Замінити нульові елементи на середнє арифметичне найменшого і найбільшого елементів
29	12	дійсний	Поміняти місцями мінімальні елементи першої та другої половин масиву
30	16	цілий	Поміняти місцями перший додатний елемент з першим від’ємним. Якщо жодного від’ємного (чи то додатного) елемента в масиві немає, видати про це повідомлення

Лабораторна робота № 12
Двовимірні масиви

Мета роботи: набути практичних навиків програмного опрацювання двовимірних масивів засобами Visual C++.

Теоретичні відомості

Організація багатовимірних масивів

Вимірність масиву визначається кількістю індексів. Елементи одновимірного масиву (вектора) мають один індекс, двовимірного масиву (матриці, таблиці) – два індекси: перший з них – номер рядка, другий – номер стовпця. Кількість індексів у масивах є необмежена. При розміщуванні елементів масиву в пам'яті комп'ютера першою чергою змінюється крайній правий індекс, потім решта – справа наліво.

Багатовимірний масив оголошується у програмі в такий спосіб:

<тип> *<ім'я>* [*<розмір1>*] [*<розмір2>*] … [*<розмірN>*];

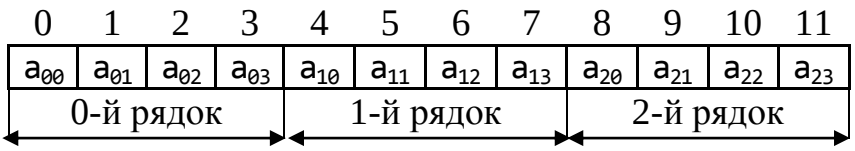
Кількість елементів масиву дорівнює добутку кількості елементів за кожним індексом. Наприклад, матриця з 3-х рядків і 4-х стовпців оголошена як

```
int a[3][4];
```

має 12 елементів цілого типу:

```
a[0][0], a[0][1], a[0][2], a[0][3],
a[1][0], a[1][1], a[1][2], a[1][3],
a[2][0], a[2][1], a[2][2], a[2][3];
```

Під масив надається пам'ять, потрібна для розміщення усіх його елементів. Елементи масиву один за одним, з першого до останнього, запам'ятовуються у послідовно зростаючих адресах пам'яті так само, як і елементи одновимірного масиву. Наприклад, масив `int a[3][4]` зберігається у пам'яті у такий спосіб:



Наведемо ще кілька прикладів оголошення масивів:

```
int x[5][5];     // Матриця 5×5=25 елементів цілого типу
char S[10][3]; // Двовимірний масив з 10×3=30  елементів символьного типу
double z[4][5][4]; // Тривимірний масив з 4×5×4=80 елементів дійсного типу
```

Введення-виведення двовимірних масивів

Здійснювати введення-виведення значень елементів масиву можна лише поелементно, для чого слід організовувати цикли, в яких послідовно змінюватимуться значення індексів елементів.

1) Введення-виведення матриць у консольному режимі

а) **Введення у консолі** матриці a розміром 3×4:

1 спосіб (див. приклад 1):

```
for(i=0; i<3; i++)
for(j=0; j<4; j++)
{ cout<<"Введіть елемент "<<i+1<<"-го рядка "<<j+1<<"-го стовпця: ";
  cin>>a[i][j];
}
```

2 спосіб:

```
for(i=0; i<3; i++)
{ cout<<"Введіть 4 елементи "<<i+1<<"-го рядка "<<endl;
  for(j=0; j<4; j++) cin>>a[i][j];
}
```

3 спосіб (див. приклад 2):

```
cout<<"Введіть матрицю з 3-х рядків і 4-х стовпців:"<<endl;
for(i=0; i<3; i++)
for(j=0; j<4; j++) cin>>a[i][j];
```

б) Для **виведення у консолі** двовимірного масиву a розміром 3×4 у вигляді матриці елементи слід розмістити у рівні стовпці. Це можна зробити за допомогою символу табуляції "\t":

```
for(i=0; i<3; i++)
{
  for(j=0; j<4; j++) cout << a[i][j] << "\t";
  cout << endl;
}
```

При виведенні елементів матриць дійсних чисел командою cout<< доречним є використання маніпуляторів форматування:

– **setprecision**, який обмежує кількість знаків після десяткової крапки;

– **fixed**, який задає формат з фіксованою крапкою;

– **setw**, який дозволяє задавати ширину (мінімальну кількість символьних позицій) кожного виведеного числа. Якщо виведене число має ширину меншу, аніж зазначену у модифікаторі setw, перед ним будуть виведені пробіли:

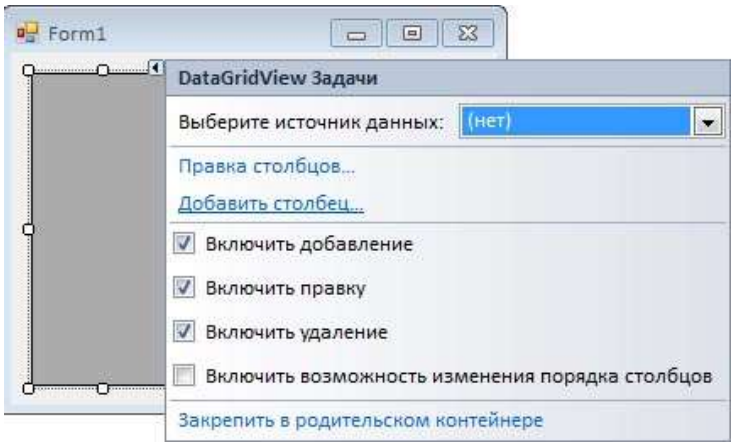
```
for(i=0; i<3; i++)
{ for(j=0; j<6; j++)
    cout<<fixed<<setprecision(2)<<setw(5)<<a[i][j]/3.<<"\t";
  cout << endl;
}
```

Для можливості використання цих маніпуляторів слід долучити заголовний файл iomanip командою #include <iomanip>.

2) Використання елемента dataGridView

Елемент DataGridView має вигляд таблиці з клітинками, в кожній з яких можна розмістити величину рядкового типу (String^). Щоб звернутись до значення Value конкретної клітинки слід зазначити номер рядка Rows[i] і номер стовпця Cells[j]: dataGridView1->Rows[i]->Cells[j]->Value.

Після розміщення на формі елемента `DataGridView` через діалогове вікно *DataGridView Задачи*, яке автоматично з'явиться, слід командою *Добавить столбец* утворити чотири стовпця, натиснувши чотири рази кнопку *Добавить*.



За допомогою команди *Правка столбцов* діалогового вікна *DataGridView Задачи* можна налаштувати параметри кожного зі стовпців, приміром зменшити ширину (`Width`) стовпців від 100 за замовчуванням до 50 пікселів.

Створювати (додавати) нові рядки будемо у програмному коді, створивши функцію `Form1_Load` і вписавши в неї команду створення трьох рядків:

```
dataGridView1->Rows->Add(3);
```

Тепер доцільно встановити значення `False` для таких властивостей `DataGridView`:

– `ColumnHeadersVisible` (заголовок стовпця);

– `RowHeadersVisible` (стовпець із заголовками рядків);

– `AllowUserToAddRows` (дозвіл на автоматичне долучення рядків).

а) **Введення** матриці дійсних чисел `a` розміром `3×4` з `dataGridView`:

```
double a[3][4];
for (int i=0; i<3; i++)
for (int j=0; j<4; j++)
  a[i][j]=Convert::ToDouble(dataGridView1->Rows[i]->Cells[j]->Value);
```

або

```
for (int i=0; i<3; i++)
for (int j=0; j<4; j++)
  a[i][j]=Convert::ToDouble(dataGridView1[j,i]->Value);
```

б) **Виведення** матриці `a` розміром 3×4 **на форму** до `dataGridView`:

```
for (int i=0; i<3; i++)
for (int j=0; j<4; j++)
  dataGridView1[j,i]->Value = Convert::ToString(a[i][j]);
```

в) **Очищення** `dataGridView`:

```
for (int i=0; i<3; i++)
for (int j=0; j<4; j++) dataGridView1[j,i]->Value = "";
```

г) **Заповнення** `dataGridView` випадковими числами:

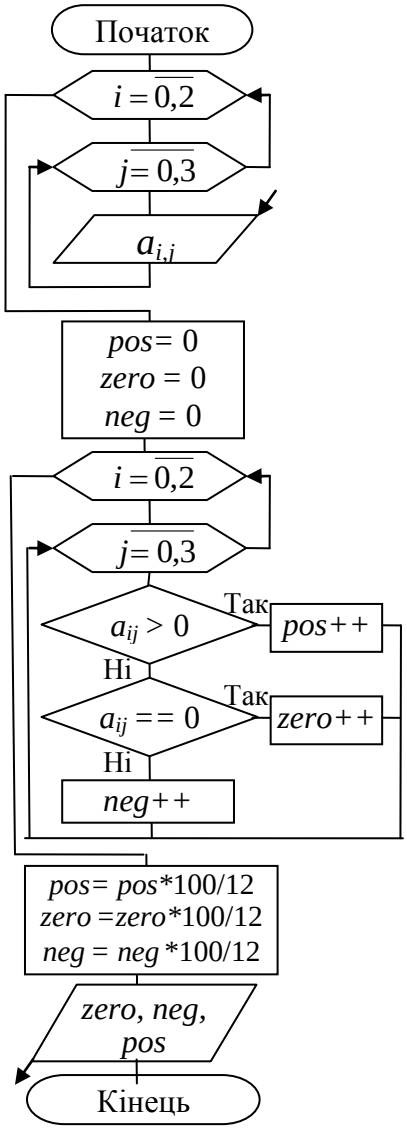
```
Random^ rnd = gcnew Random();
for (int i=0; i<3; i++)
for (int j=0; j<4; j++)
  dataGridView1[j,i]->Value = Convert::ToString(rnd->Next(-100,101)/10.);
```

Використаний тут метод `Next(-100,101)` генерує ціле число в діапазоні від -100 до 100, а за рахунок ділення цього числа на дійсне значення 10.0 клітинки `dataGridView1` будуть заповнені випадковими дійсними числами від -10 до 10. Крім методу `Next()`, для типу `Random` існує метод `NextDouble()`, який генерує випадкове число в діапазоні від 0 до 1.

Приклади програм

Приклад 1. Ввести матрицю 3×4 цілих чисел та обчислити процентний вміст від'ємних, нульових і додатних елементів.

Текст програмного коду та схема алгоритму:




```
#include <iostream>
//бібліотека для використання маніпуляторів форматування
#include <iomanip>
using namespace std;

int main()
{ setlocale(0,".1251");
  int a[3][4],i,j;
  for(i=0; i<3; i++)
  for(j=0; j<4; j++)
  { cout<<"Введіть елемент "<<i+1<<"-го рядка "
        <<j+1<<"-го стовпця: ";
    cin>>a[i][j];
  }
  double pos,zero,neg;
  pos=zero=neg=0;
  for(i=0; i<3; i++)
  for(j=0; j<4; j++)
  { if(a[i][j]>0) pos++;
    else
      if(a[i][j]==0) zero++;
      else neg++;
  }
  pos *= 100.0/12;
  neg *= 100.0/12;
  zero *= 100.0/12;
  cout<<"\nПроцентний вміст від'ємних елементів - "
      <<setprecision(3) <<neg<<" %"<<endl;
  cout<<"Процентний вміст нульових елементів - "<<zero<<" %"<<endl;
  cout<<"Процентний вміст додатних елементів - "<<pos<<" %"<<endl;
  system ("pause>>void");
  return 0;
}
```

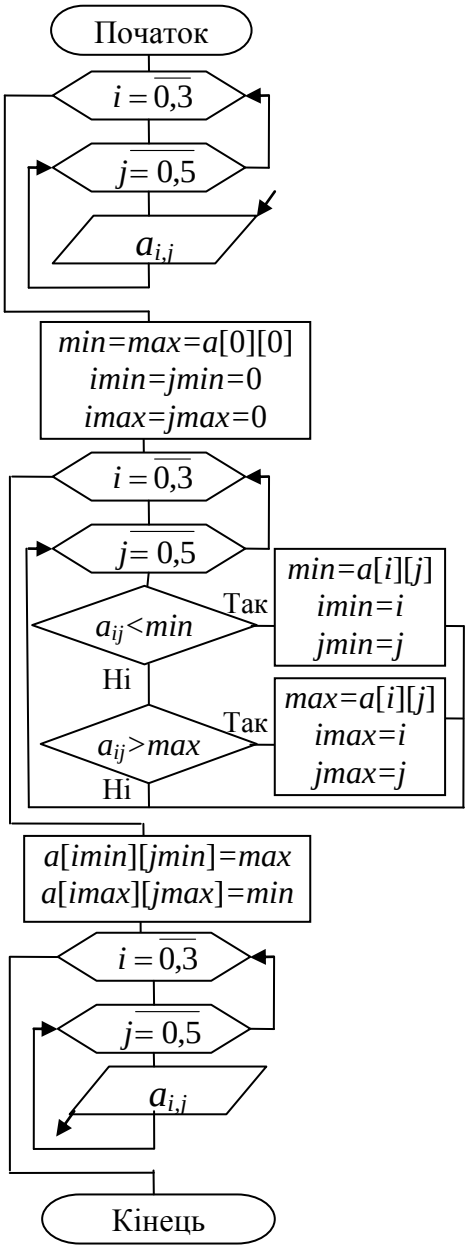
Результати роботи консольного додатка:

```
Введіть елемент 1-го рядка 1-го стовпця: 23
Введіть елемент 1-го рядка 2-го стовпця: 4
Введіть елемент 1-го рядка 3-го стовпця: 6
Введіть елемент 1-го рядка 4-го стовпця: 0
Введіть елемент 2-го рядка 1-го стовпця: -4
Введіть елемент 2-го рядка 2-го стовпця: -8
Введіть елемент 2-го рядка 3-го стовпця: 23
Введіть елемент 2-го рядка 4-го стовпця: 2
Введіть елемент 3-го рядка 1-го стовпця: 0
Введіть елемент 3-го рядка 2-го стовпця: -12
Введіть елемент 3-го рядка 3-го стовпця: 1
Введіть елемент 3-го рядка 4-го стовпця: 4

Процентний вміст від'ємних елементів - 25 %
Процентний вміст нульових елементів -  16.7 %
Процентний вміст додатних елементів -  58.3 %
```

Приклад 2. Ввести матрицю 4×6 дійсних чисел і поміняти місцями максимальний та мінімальний елементи.

Програмний код та схема алгоритму:

```
#include <iostream>
using namespace std;
int main()
{ setlocale(0,".1251");
  double a[4][6],min,max;
  int i,j,imin,jmin,imax,jmax;
  cout<<"Введіть матрицю з 4-х рядків і
        6-х стовпців:"<<endl;
  for(i=0; i<4; i++)
  for(j=0; j<6; j++) cin>>a[i][j];
  min=max=a[0][0];
  imin=jmin=imax=jmax=0;
  for(i=0; i<4; i++)
  for(j=0; j<6; j++)
  { if(a[i][j]<min)
     { min=a[i][j]; imin=i; jmin=j; }
    if(a[i][j]>max)
     { max=a[i][j]; imax=i; jmax=j; }
  }
  a[imin][jmin]=max;  a[imax][jmax]=min;
  cout<<"\nМатриця, в якій поміняні місцями
  максимальний і мінімальний елементи:"<<endl;
  for(i=0; i<4; i++)
  { for(j=0; j<6; j++) cout<<a[i][j]<<"\t";
    cout << endl;
  }
  system ("pause>>void");
  return 0;
}
```



Результати роботи консольного додатка:

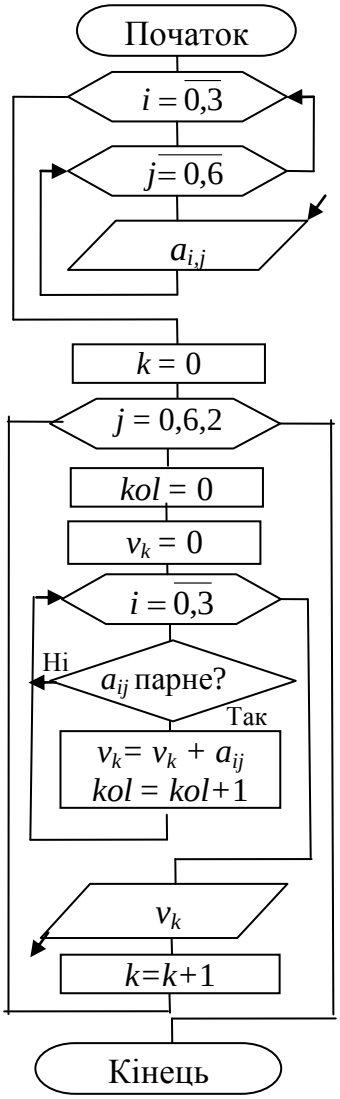
```
Введіть матрицю з 4-х рядків і 6-х стовпців:
23      0       -4      9       2.6     8.1
1       -100    4.6     7       0       9
-3      0       5.7     -0.6    34      10
1       2       3       -4      0       59

Матриця, в якій поміняні місцями максимальний і мінімальний елементи:
23      0       -4      9       2.6     8.1
1       59      4.6     7       0       9
-3      0       5.7     -0.6    34      10
1       2       3       -4      0       -100
```

Приклад 3. З елементів матриці розміром 4×7 цілих чисел сформувати вектор середніх арифметичних парних елементів непарних (1, 3, 5 і 7) стовпців матриці.

Розв'язок. Оскільки при формуванні вектора його індекси не співпадають з індексами непарних стовпців матриці, по яких слід формувати вектор, виникає потреба у використанні додаткової змінної, приміром k для індексів вектора.

Текст програмного коду та схема алгоритму:

```
#include <iostream>
using namespace std;
int main()
{ setlocale(0,".1251");
  int i,j;   int a[4][7]; double v[4];
  cout<<"Введіть матрицю з 4-х рядків
        і 7-ми стовпців:"<<endl;
  for(i=0; i<4; i++)
  for(j=0; j<7; j++) cin>>a[i][j];
  cout<<"\nВектор середніх арифметичних парних
  елементів непарних (1, 3, 5 та 7) стовпців:"<<endl;
  int kol, k=0;
  for(j=0; j<7; j+=2)
  { kol = 0;
    v[k] = 0;
    for(i=0; i<4; i++)
     if(a[i][j]%2 == 0 && a[i][j])
       { v[k] += a[i][j];
         kol++;
       }
    if(kol) v[k] /= kol;
    cout<< v[k]<<"\t\t";
    k++;
   }
  cout<<endl;
  system ("pause>>void");
  return 0;
}
```



Результати роботи консольного додатка:

```
Введіть матрицю з 4-х рядків і 7-ми стовпців:
1       3       -6      0       9       61      8
2       8       -4      45      3       -1      0
4       -45     13      4       13      8       4
20      6       1       2       7       5       1

Вектор середніх арифметичних парних елементів непарних (1, 3, 5 та 7) стовпців:
8.66667         -5          0           6
```

Приклад 4. За елементами матриці розміром 5×10 цілих чисел сформувати вектор максимальних елементів стовпців матриці.

Розв'язок. Налаштувати форму рекомендуємо у такій послідовності.

1) Розмістити на формі перший елемент dataGridView і сформувати (команда *Добавить столбец* в контекстному меню) на ньому 10 стовпців, для кожного з яких командою контекстного меню *Правка столбцов* задати для властивості Width (ширина стовпця) значення 40 пікселів (замість 100).

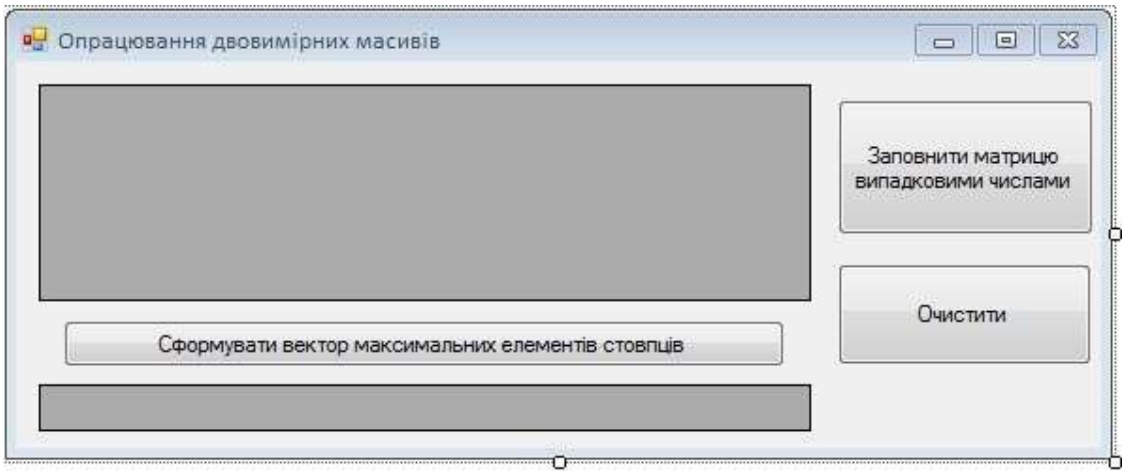
2) Задати для властивості AllowUserToAddRows (дозвіл автоматичного додавання рядків) значення False.

3) Задати для властивостей RowHeadersVisible і ColumnHeadersVisible (відображення заголовка рядків і стовпців) значення False.

4) Відредагувати розміри (властивість Size) на значення 410; 115.

5) Скопіювати (*Ctrl+C*) налаштований dataGridView1, виконати команду вставки – *Ctrl+V* (при цьому автоматично створить dataGridView2 з уже встановленими властивостями). Можна перемістити dataGridView2 під dataGridView1 і зменшити розмір dataGridView2 (властивість Size) до 410; 25.

6) Розмістити на формі три кнопки buttonта задати надписи на них (властивість Text): для button1 – *Сформувати вектор максимальних елементів стовпців*, для button2 – *Заповнити матрицю випадковими числами*, для button3 – *Очистити*. Задати надпис на формі (властивість Text) – *Опрацювання двовимірних масивів*.



Подвійним клацанням по відповідних кнопках на формі створити функції Form1_Load, button1_Click, button2_Click і button3_Click та вписати в них програмний код.

Текст програмного коду:

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{  dataGridView1->Rows->Add(5); dataGridView2->Rows->Add();
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
 int a[5][10], x[10];
 for (int i=0; i<5; i++)
 for (int j=0; j<10; j++)
   a[i][j]=Convert::ToInt32(dataGridView1->Rows[i]->Cells[j]->Value);
 for (int j=0; j<10; j++)
 { x[j]=a[0][j];
   for (int i=0; i<5; i++)
     if(x[j]<a[i][j]) x[j]=a[i][j];
   dataGridView2->Rows[0]->Cells[j]->Value = Convert::ToString(x[j]);
 }
}
private: System::Void button2_Click(System::Object^ sender,System::EventArgs^ e)
{  Random^ rnd = gcnew Random();
   for (int i=0; i<5; i++)
   for (int j=0; j<10; j++) dataGridView1[j,i]->Value = rnd->Next(50);
}
private: System::Void button3_Click(System::Object^ sender,System::EventArgs^ e)
{ for (int i=0; i<5; i++)
  for (int j=0; j<10; j++)  dataGridView1[j,i]->Value = "";
  for (int j=0; j<10; j++)  dataGridView2[j,0]->Value = "";
}
```

Результати роботи:

Питання та завдання для самоконтролю

1) Які з наведених оголошень двовимірних масивів неправильні й чому?
   а) `int C[1..5, 1..5];` в) `double C[1..5][1..5];`
   б) `double C[5][5];` г) `int C: [5][5];`
2) В який спосіб розміщуються елементи двовимірних масивів в оперативній пам'яті?
3) Запишіть оператор оголошення матриці цілих чисел `S` розміром 7×3.
4) Під яким номером записано оператор, що обчислює суму елементів головної діагоналі матриці `A` цілих чисел розміром 5×5?
   а) `for(i=0, s=0; i<5; i++) s++;`
   б) `for(i=0, s=0; i<5; i++) s+=A[i][i];`
   в) `for(i=0, s=0; i<5; i++) for(j=0; j<5; j++) s+=A[i][j];`
   г) `for(i=0, s=0; i<5; i++) A[i][i]=0;`
5) Чи можна виконувати опрацювання двовимірного масиву, організувавши зовнішній цикл по стовпцях, а внутрішній – по рядках?

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи скласти схеми та написати програми мовою С++ для розв'язання завдань, поданих в табл. 12.1 … 12.3 відповідно до індивідуального варіанта.

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 12.1

Індивідуальні завдання середнього рівня складності

№ вар.	Індивідуальне завдання
1	У матриці дійсних чисел з 5-ти рядків і 4-х стовпців обчислити кількість додатних, від'ємних і нульових елементів
2	У матриці цілих чисел розміром 4×5 визначити найбільший елемент та його індекси
3	Визначити мінімальний елемент головної діагоналі квадратної матриці розміром 5×5 і номер рядка, в якому він міститься
4	Поміняти місцями елементи першої рядка матриці дійсних чисел розміром 4×4 з елементами її неголовної діагоналі
5	У матриці цілих чисел розміром 3×5 замінити від'ємні елементи нулями
6	Визначити максимальний і мінімальний елементи матриці дійсних чисел розміром 6×6
7	Обчислити вектор середньоарифметичних значень елементів рядків матриці дійсних чисел розміром 5×4
8	Обчислити суму елементів неголовної діагоналі матриці 5×5 цілих чисел

Закінчення табл. 12.1

№ вар.	Індивідуальне завдання
9	Для матриці цілих чисел розміром 5×5 обчислити транспоновану матрицю
10	Визначити номер стовпця матриці дійсних чисел розміром 3×6 з найменшим елементом
11	Визначити мінімальний елемент неголовної діагоналі матриці цілих чисел розміром 5×5 і номер стовпця, в якому він розміщений
12	Обчислити вектор сум елементів рядків матриці цілих чисел розміром 7×3
13	Замінити в непарних рядках матриці дійсних чисел розміром 7×4 від'ємні елементи на нулі, а додатні елементи – на одиниці
14	Обчислити суми елементів головної і неголовної діагоналей матриці дійсних чисел розміром 5×5 та різницю між цими сумами
15	У матриці дійсних чисел розміром 7×5 обчислити суму всіх від'ємних елементів перших чотирьох рядків
16	У матриці цілих чисел розміром 4×5 замінити всі від'ємні елементи на нулі
17	Обчислити добуток мінімального елемента матриці цілих чисел розміром 4×5 на значення середнього арифметичного матриці
18	У матриці цілих чисел розміром 6×4 обчислити середнє арифметичне додатних елементів
19	У матриці цілих чисел розміром 5×4 замінити в непарних рядках додатні елементи на 1, а в парних – від'ємні на 0
20	У матриці дійсних чисел розміром 6×3 обчислити добуток всіх від'ємних елементів у парних рядках
21	У матриці цілих чисел розміром 3×5 обчислити кількість елементів, менших за середнє арифметичне
22	У матриці дійсних чисел розміром 5×3 замінити всі елементи, які більші 2.5, на –1
23	Обчислити вектор сум модулів елементів рядків матриці дійсних чисел розміром 4×5
24	У матриці цілих чисел розміром 7×4 визначити найменший елемент з числа додатних і найбільший з числа від'ємних і поміняти їх місцями
25	Обчислити вектор елементів головної діагоналі матриці дійсних чисел розміром 5×5
26	Обчислити вектор сум квадратів елементів стовпців матриці дійсних чисел розміром 3×5
27	У матриці цілих чисел розміром 5×5 поміняти місцями елементи головної та неголовної діагоналей
28	У матриці цілих чисел розміром 5×5 замінити всі парні елементи на нулі
29	Обчислити різницю сум елементів першого рядка й останнього стовпця матриці дійсних чисел розміром 4×6
30	Обчислити вектор сум елементів головної і неголовної діагоналей матриці дійсних чисел розміром 6×6

Таблиця 12.2

Індивідуальні завдання базового рівня складності

№ вар.	Індивідуальне завдання
1	Обчислити вектор сум елементів непарних стовпців матриці 3×7 цілих чисел
2	Обчислити вектор скалярних добутків рядків матриці дійсних чисел розміром 4×4 на її останній стовпець
3	Обчислити вектор добутків непарних елементів парних рядків матриці цілих чисел розміром 6×4
4	Обчислити вектор скалярних добутків елементів першого рядка матриці цілих чисел розміром 4×4 на стовпці цієї матриці
5	Обчислити визначник введеної матриці 3×3 з дійсних чисел
6	Обчислити вектор як стовпець матриці дійсних чисел розміром 6×4 з найменшою сумою елементів
7	Обчислити вектор добутків парних елементів непарних стовпців матриці цілих чисел розміром 4×5
8	Обчислити вектор як рядок матриці цілих чисел розміром 4×5 з найбільшою сумою елементів
9	Обчислити вектор сум непарних елементів парних рядків матриці цілих чисел розміром 6×6
10	Обчислити вектор скалярних добутків елементів стовпців матриці дійсних чисел розміром 3×3 на її головну діагональ
11	Обчислити вектор з найменших додатних елементів стовпців матриці дійсних чисел розміром 4×6
12	Замінити кутові елементи матриці дійсних чисел розміром 5×6 на значення середнього арифметичного елементів
13	Замінити елементи головної діагоналі матриці цілих чисел розміром 5×5 сумами елементів стовпців
14	Обчислити вектор середньоарифметичних додатних елементів парних рядків матриці цілих чисел розміром 7×8
15	Обчислити вектор з найбільших елементів рядків матриці дійсних чисел розміром 7×6
16	Замінити елементи неголовної діагоналі матриці цілих чисел розміром 4×4 сумами елементів її рядків
17	Обчислити вектор скалярних добутків рядків матриці цілих чисел розміром 5×5 на її неголовну діагональ
18	Обчислити вектор як рядок матриці, який містить найбільший елемент матриці цілих чисел розміром 5×6
19	Замінити елементи головної діагоналі матриці цілих чисел розміром 6×6 сумою максимального і мінімального елементів матриці
20	Замінити елементи неголовної діагоналі матриці дійсних чисел розміром 4×4 значенням мінімального елемента матриці

Закінчення табл. 12.2

№ вар.	Індивідуальне завдання
21	Замінити нульові елементи матриці цілих чисел розміром 5×5 її максимальним елементом
22	У матриці дійсних чисел розміром 7×3 поміняти місцями перший і останній від'ємні елементи
23	Обчислити вектор сум додатних елементів рядків матриці цілих чисел розміром 6×5. Упорядкувати цей вектор за зростанням
24	З-посеред рядків матриці цілих чисел розміром 5×4 знайти той, для якого сума непарних елементів буде мінімальною, і побудувати з цього рядка вектор
25	У матриці дійсних чисел 6×4 обчислити суму додатних і суму від'ємних елементів, замінити кутові елементи на значення більшої за модулем суми
26	Обчислити вектор скалярних добутків стовпців матриці дійсних чисел розміром 4×4 на її неголовну діагональ
27	Замінити від'ємні елементи матриці дійсних чисел розміром 4×6 на значення середнього арифметичного додатних елементів
28	Обчислити вектор максимальних елементів рядків матриці дійсних чисел розміром 7×5
29	Обчислити вектор середньоарифметичних першого і останнього елементів парних рядків матриці цілих чисел розміром 7×8
30	Обчислити вектор квадратів елементів мінімальних елементів непарних стовпців матриці дійсних чисел розміром 6×5

Таблиця 12.3

Індивідуальні завдання високого рівня складності

№ вар.	Індивідуальне завдання
1	По введеній матриці дійсних чисел розміром 5×7 обчислити нову матрицю, в якій кожен елемент обчислюється як півсума середньоарифметичних відповідних рядка і стовпця
2	У матриці дійсних чисел розміром 5×5 обчислити визначник
3	Найменший за довжиною рядок матриці дійсних чисел розміром 5×4 замінити на найбільший за довжиною
4	Обчислити добуток матриці 5×5 цілих чисел на її транспоновану матрицю
5	У матриці дійсних чисел розміром 7×3 обчислити номер рядка, довжина якого (як вектора) є максимальною
6	Створити вектор з рядка матриці дійсних чисел розміром 8×3, найменш віддаленого від другого рядка (відстань між рядками обчислюється за формулою $d_i = \sum_{j=0}^{2} \left| a_{ij} + a_{2j} \right|$)
7	Ввести матрицю 5×5 цілих чисел від 0 до 9. Якщо кількість повторів елемента матриці збігається з самим елементом, то замінити його на нуль

Продовження табл. 12.3

№ вар.	Індивідуальне завдання
8	Упорядкувати за зростанням (зліва направо) елементи всіх рядків матриці дійсних чисел розміром 4×4, а тоді за зростанням (зверху вниз) – елементи всіх стовпців
9	Сформувати вектор як рядок матриці дійсних чисел розміром 5×5, найбільш віддалений від першого рядка (відстань від першого рядка матриці до *i*-го обчислюється за формулою $d_i = \sum_{j=0}^{4} \lvert a_{ij} \rvert + \lvert a_{1j} \rvert,\ i \neq 1$)
10	Сформувати вектор з рядка матриці дійсних чисел розміром 6×4 з найменшою сумою $\sum_{j=0}^{3} a_{ij}^2$
11	Обчислити вектор як стовпець матриці дійсних чисел розміром 3×7 з найбільшою вагою (вага стовпця матриці обчислюється $W_j = \sum_{i=0}^{2} \lvert a_{ij} \rvert$)
12	Сформувати вектор як стовпець матриці дійсних чисел розміром 8×6 з найбільшою сумою $\sum_{j=0}^{5} \lvert a_{ij} \rvert + a_{ij}$
13	Кожен від'ємний елемент матриці дійсних чисел розміром 4×8 замінити сумою додатних елементів того рядка, в якому розміщений цей елемент
14	Сформувати вектор як стовпець матриці дійсних чисел розміром 6×8, найменш віддалений від першого (відстань між *j*-им стовпцем і першим обчислюється за формулою $d_j = \sum_{i=0}^{5} \lvert a_{ij} \rvert \cdot \lvert a_{i1} \rvert,\ j \neq 1$)
15	Обчислити вектор як рядок матриці дійсних чисел розміром 5×3 з найменшою вагою (вага рядка матриці обчислюється за формулою $W_i = \sum_{j=0}^{2} \lvert a_{ij} \rvert$)
16	У матриці цілих чисел розміром 6×6 визначити мінімальний елемент в секторі над головної діагоналлю і мінімальний елемент в секторі під головною діагоналлю. На більше з цих значень замінити елементи головної діагоналі
17	Ввести матрицю розміром 3×5 з цілих чисел від 0 до 9. Обчислити процентний вміст кожного з цих чисел у матриці
18	Обчислити вектор як стовпець матриці дійсних чисел розміром 3×6 з найменшим значенням $W_j = \sqrt{\sum_{i=0}^{5} a_{ij}^2}$
19	Визначити координати елемента матриці дійсних чисел розміром 5×7 з найменшою вагою (вага обчислюється за формулою $W_{ij} = \sum_{i=0}^{4} \sum_{j=0}^{6} \left\lvert \frac{a_{ij}}{i+j} \right\rvert$)
20	У матриці цілих чисел розміром 6×7 визначити рядок з мінімальною сумою і поміняти місцями цей рядок з першим

Закінчення табл. 12.3

№ вар.	Індивідуальне завдання
21	Створити вектор з рядка матриці дійсних чисел розміром 7×5, найбільш віддаленого від третього рядка (відстань між рядками обчислюється за формулою $d_i = \sum_{j=0}^{4} \lvert a_{ij} + a_{3j} \rvert$)
22	Ввести матрицю розміром 6×4 з цілих чисел від 0 до 9. Обчислити вектор з 10-ти елементів як значення кількості повторів цих констант у матриці
23	Якщо в матриці дійсних чисел 4×5 сума додатних чисел більше модуля суми від'ємних, то замінити кутові елементи середнім арифметичним елементів матриці. Інакше видати про це повідомлення
24	Обчислити вектор як найбільш віддалений стовпець від (n – 1)-го стовпця матриці дійсних чисел розміром 6×5 (відстань обчислюється за формулою $d_j = \sum_{i=0}^{4} \lvert a_{ij} \rvert + \lvert a_{i,n-1} \rvert$). Значення n, як і значення елементів матриці, ввести з екрану
25	Ввести дві матриці дійсних чисел 4×5. Поміняти місцями рядки матриць, які містять максимальні елементи
26	Ввести матрицю дійсних чисел 5×5 і вектор з п'яти дійсних чисел. Замінити всі рядки матриці, в яких є від'ємні числа, на елементи вектора
27	За матрицею цілих чисел розміром 4×5 обчислити вектор як середнє арифметичне стовпців, які містять максимальний і мінімальний елементи
28	За матрицею дійсних чисел 4×5 сформувати нову матрицю зі значень ваги відповідних елементів (вага обчислюється за формулою $W_{ij} = \sum_{i=0}^{3} \sum_{j=0}^{4} \left\lvert \frac{a_{ij}}{i+j} \right\rvert$)
29	Якщо в матриці цілих чисел 5×6 міститься парна кількість від'ємних елементів, то замінити будь-яку половину з цих чисел на нуль, інакше замінити всі від'ємні елементи на значення їхньої кількості
30	Вивести елемент матриці 6×6 з найбільшою відстанню до діагоналі (відстань обчислюється за формулою $d_{ij} = \sum_{i=0}^{5} \left\lvert \sum_{j=0}^{5} \lvert a_{ij} - a_{jj} \rvert - a_{ii} \right\rvert$)

Лабораторна робота № 13

Опрацювання двовимірних масивів у функціях

Мета роботи: набути практичних навиків програмного опрацювання елементів матриць у функціях засобами Visual C++.

Теоретичні відомості

При передаванні до функції *багатовимірних масивів* усі розмірності, якщо вони є невідомі на етапі компіляції, мають передаватися як параметри. Наприклад, заголовок функції, яка обчислює суму елементів динамічного двовимірного масиву, може мати вигляд:

```
int sum(int **a, int m, int n);
```

Виклик цієї функції у програмі може бути таким:

```
cout << "Сума елементів a: " << sum((int**)a,m,n);
```

Для звичайного статичного двовимірного масиву, коли обидві розмірності є відомі та є константами, заголовок функції матиме вигляд

```
int sum(int a[4][6]);
```

При опрацюванні у функціях як одновимірних, так і двовимірних масивів, які є параметрами цих функцій, насправді до функції передаються не самі масиви, а вказівники на їхні перші елементи.

Розглянемо кілька варіантів передавання матриці цілих чисел `a[3][4]` до функції `print()`, яка організовує виведення цієї матриці на екран у консолі.

1) Якщо розміри обох індексів є відомі, то проблем немає:

```
void print(int a[3][4])
{ for (int i=0; i<3; i++)
  { for (int j=0; j<4; j++) cout << a[i][j] << "\t";
    cout << endl;
} }
```

Матриця і тут передається як вказівник, а розмірності наведено просто для повноти опису. Виклик такої функції може бути таким:

```
int x[3][4];
. . . . . .
print(x);
```

2) Перша розмірність для обчислення адреси елемента є неважлива, тому її можна передавати як параметр:

```
void print(int a[][4], int m)
{ for (int i=0; i<m; i++)
  { for (int j=0; j<4; j++) cout << a[i][j] << "\t";
    cout << '\n';
} }
```

Виклик цієї функції:

```
int x[3][4];
. . . . . .
print(x,3);
```

3) Найскладніший випадок – коли треба передавати обидві розмірності. Наведений нижче код функції є поширеною помилкою:

```
void print(int a[][], int m, int n)  // Помилка!
{ for (int i=0; i<m; i++)
  { for (int j=0; j<n; j++) cout << A[i][j] << "\t";
    cout << '\n';
} }
```

По-перше, опис параметра `a[][]` є неприпустимий, оскільки для обчислення адреси елемента двовимірного масиву слід знати другу розмірність. У такому разі найбільш поширеним і грамотним є застосування динамічних масивів, створювання яких більш докладно розглянуто в подальших лабораторних і практичних заняттях.

Приклади програм

Приклад 1. Ввести матрицю дійсних чисел розмірності 4×3 й обчислити за допомогою функції мінімальний елемент матриці та його індекси.

Розв'язок. Оскільки функція має обчислити і повернути три значення, то один з результатів – мінімальний елемент – буде повернуто в головну програму через оператор `return`, а інші два результати – індекси – передаватимуться за посиланням (чи за вказівником).

Схеми алгоритму функції та основної програми:

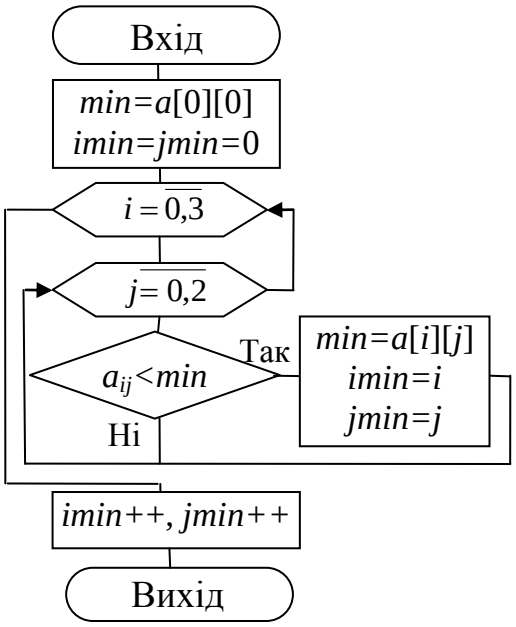


Схема алгоритму функції **MIN ()**

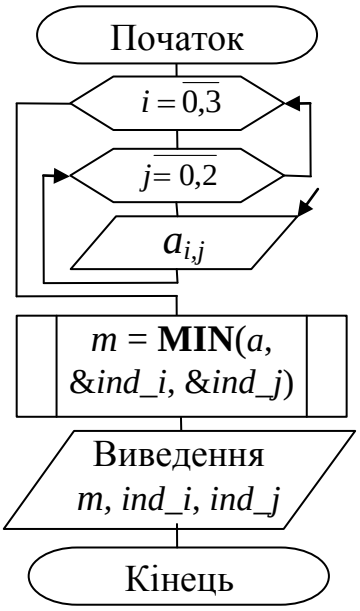


Схема алгоритму до основної програми

Тексти функції та її виклику в основній програмі:

```
#include <iostream>
using namespace std;

double MIN(double a[4][3], int& imin, int& jmin)
{
  double min=a[0][0];
  imin=jmin=0;
  for(int i=0; i<4; i++)
  for(int j=0; j<3; j++)
   if(a[i][j]<min)
   { min=a[i][j];
     imin=i;  jmin=j;
   }
  imin++; jmin++;
  return min;
}
int main ()
{
  setlocale(0,".1251");
  double a[4][3], m;
  int i, j, ind_i, ind_j;
  cout<<" Введіть матрицю з 4-х рядків і 3-х стовпців:"<<endl;
  for(i=0; i<4; i++)
  for(j=0; j<3; j++) cin>>a[i][j];
  m = MIN(a, ind_i, ind_j);
  cout<< "\n Мінімальний елемент " << m << "\n розміщений у "
      << ind_i <<"-му рядку і " << ind_j <<"-му стовпці" << endl;
  system ("pause>>void");
  return 0;
}
```

Результати роботи консольного додатка:

```
Введіть матрицю з 4-х рядків і 3-х стовпців:
34      0       4.3
12.7    -7.1    0.3
-5      9       -1
2       -9.4    1.5

Мінімальний елемент -9.4
розміщений у 4-му рядку і 2-му стовпці
```

Приклад 2. Ввести матрицю дійсних чисел розмірності 5×5 і за допомогою функції замінити елементи головної діагоналі на середнє арифметичне відповідного рядка.

Схеми алгоритму функції та основної програми:

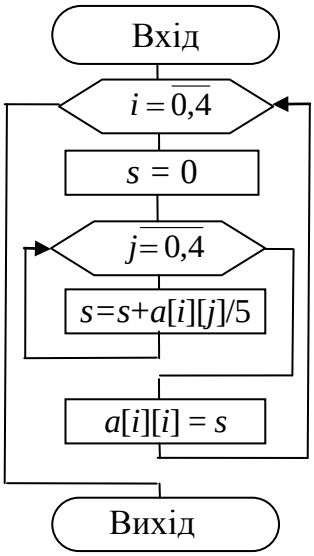


Схема алгоритму функції **zamina ()**

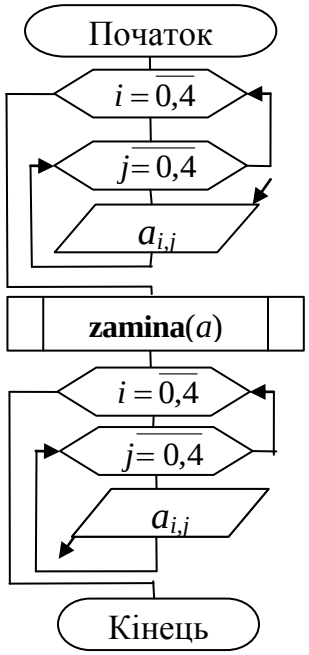


Схема алгоритму до основної програми

Текст функції та основної програми:

```
#include <iostream>
using namespace std;

void zamina(double a[5][5])
{ for(int i=0; i<5; i++)
  { double s=0;
    for(int j=0; j<5; j++) s+=a[i][j];
    a[i][i]=s/5;
} }
int main ()
{ setlocale(0,".1251");
  double a[5][5];
  int i, j;
  cout<<"Введіть матрицю з 5-ти рядків і 5-ти стовпців:"<<endl;
  for(i=0; i<5; i++)
  for(j=0; j<5; j++) cin >> a[i][j];
  zamina(a);
  cout<< "\nМатриця, в якій елементи головної діагоналі поміняли
          \nна середнє арифметичне відповідного рядка:" << endl;
  for(i=0; i<5; i++)
  { for(j=0; j<5; j++) cout << a[i][j] << "\t";
    cout << endl;
  }
  system ("pause>>void");
  return 0;
}
```

Результати роботи консольного додатка:

```
Введіть матрицю з 5-ти рядків і 5-ти стовпців:
23      6       0       -6.1    45.5
-6      0       1.5     4       -1
6.7     4       99      0       1
9       11      -5      5.8     0.1
3       3.2     7       49      9

Матриця, в якій елементи головної діагоналі поміняли
на середнє арифметичне відповідного рядка:
13.68   6       0       -6.1    45.5
-6      -0.3    1.5     4       -1
6.7     4       22.14   0       1
9       11      -5      4.18    0.1
3       3.2     7       49      14.24
```

Приклад 3. Розробити програму для введення матриці цілих чисел розмірністю 6×3 і створити функцію обчислення елементів вектора як середні арифметичні значення непарних елементів парних рядків.

Розв'язок. У функції `Vektor()` для обчислення елементів вектора спочатку слід організувати цикл для переміщення по парних рядках (2, 4, 6). У кожному з парних рядків, рухаючись по стовпцях, підсумовуємо тільки парні елементи та їхню кількість. Перед обчисленням середнього арифметичного, тобто перед діленням, перевіряємо можливість відсутності в даному парному рядку непарних елементів, тобто виключаємо поділ на нуль. Змінна *k* є потрібна для послідовного формування індексів вектора, оскільки їхня нумерація не збігається з нумерацією рядків у матриці.

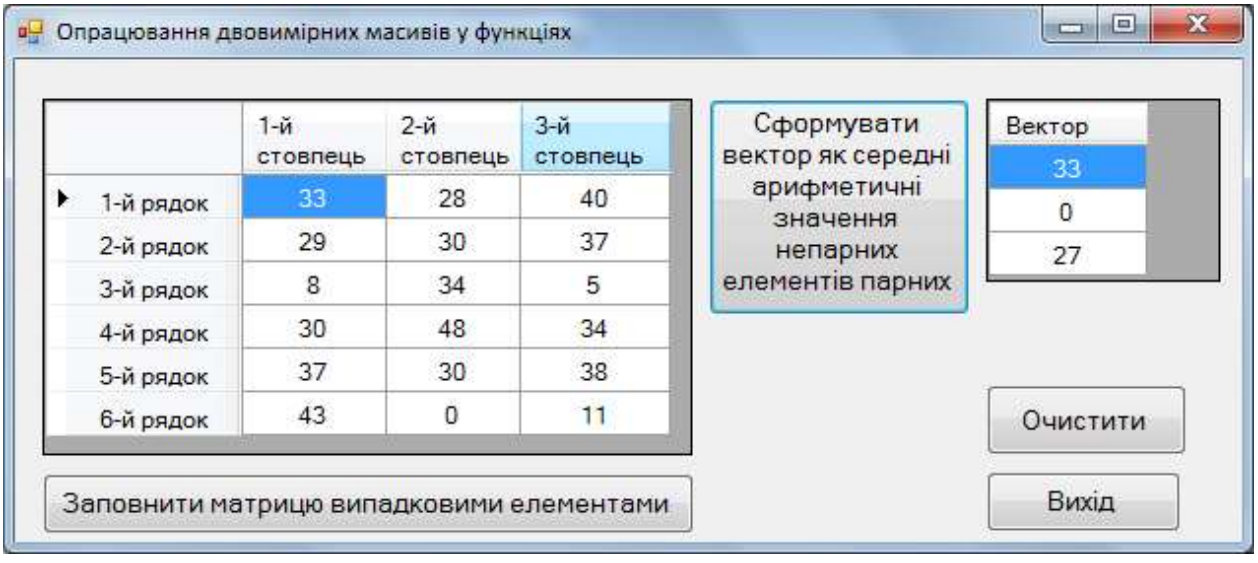
Після розміщення на формі двох елементів `dataGridView` слід створити три стовпці для `dataGridView1` та один – для `dataGridView2`. Налаштування обох `dataGridView`, на відміну від прикладу 4 у лабораторній роботі 12, організуємо в програмному коді для функції `Form1_Load`.

Текст функції та основної програми:

```
void Vektor(int a[6][3], double X[3])
{ int i, j, kol, k = 0;
  for(i=1; i<6; i+=2)
   { X[k] = kol = 0;
     for(j=0; j<3; j++)
       if(a[i][j] % 2 == 1)
        { X[k] += a[i][j]; kol++;  }
     if(kol) X[k] /= kol ;   else X[k] = 0;
     k++;
} }
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{ int i, j, a[6][3];   double V[3];
  for(i=0; i<6; i++)
  for(j=0; j<3; j++)
   a[i][j]=Convert::ToInt32(dataGridView1[j,i]->Value);
  Vektor(a, V);       // Виклик функції Vektor
```

```
  for(i=0; i<3; i++)
   dataGridView2[0,i]->Value =V[i].ToString("0.##");
}
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{ dataGridView1->AllowUserToAddRows = false;
  dataGridView2->AllowUserToAddRows = false;
  for(int i=0; i<3; i++)
    dataGridView1->Columns[i]->DefaultCellStyle->Alignment =
                              DataGridViewContentAlignment::MiddleCenter;
  dataGridView2->Columns[0]->DefaultCellStyle->Alignment =
                              DataGridViewContentAlignment::MiddleCenter;
  dataGridView1->RowHeadersWidth = 90;
  dataGridView2->RowHeadersVisible = false;
  dataGridView1->Rows->Add(6);  dataGridView2->Rows->Add(3);
  for(int i=0; i<6; i++)
   dataGridView1->Rows[i]->HeaderCell->Value = Convert::ToString(i+1)+"-й рядок";
  for(int j=0; j<3; j++)
   dataGridView1->Columns[j]->HeaderText = Convert::ToString(j+1)+"-й стовпець";
  dataGridView2->Columns[0]->HeaderText = "Вектор";
  button2_Click(button2, e); // Виклик функції button2_Click
}
private: System::Void button2_Click(System::Object^sender,System::EventArgs^e)
{ Random^ rnd = gcnew Random();
  for (int i=0; i<6; i++)
  for (int j=0; j<3; j++)
    dataGridView1->Rows[i]->Cells[j]->Value = rnd->Next(50);
}
private: System::Void button3_Click(System::Object^sender,System::EventArgs^e)
{ for (int i=0; i<6; i++)
  for (int j=0; j<3; j++) dataGridView1[j,i]->Value = "";
  for (int i=0; i<3; i++) dataGridView2[0,i]->Value = "";
}
private: System::Void button4_Click(System::Object^ sender,System::EventArgs^e)
{ Close(); }
```

Результати роботи:



Питання та завдання для самоконтролю

1) Як передаються матриці до функцій у якості аргументів?
2) Який тип матиме функція, яка повинна для матриці цілих чисел:
   а) визначити максимальний елемент?
   б) обчислити середнє арифметичне?
   в) сформувати вектор за певним правилом?
   г) змінити розміщення певних елементів у цій матриці?
3) Які з наведених заголовків функцій є помилковими і чому саме?

```
void fun(int a[][4], int m);
void fun(int a[4][], int m);
void fun(int a[3][4]);
void fun(int a[][], int m, int n);
```

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи скласти схеми алгоритмів функцій та основних програм, а також написати програмний код функцій та основних програм мовою С++ для розв'язання завдань, поданих в табл. 13.1 … 13.3 відповідно до індивідуального варіанта.

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 13.1

Варіанти завдань для створення функції з одним або двома результатами

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	5×5	цілий	Обчислити кількість від'ємних елементів матриці
2	4×4	дійсний	Обчислити суму елементів головної діагоналі матриці
3	6×4	цілий	Визначити найменший елемент матриці
4	3×3	дійсний	Обчислити добуток ненульових елементів матриці
5	4×5	цілий	Обчислити середнє арифметичне мінімального та максимального елементів матриці
6	3×5	дійсний	Обчислити кількість елементів, значення яких більше за значення першого елемента матриці
7	5×3	цілий	Обчислити середнє арифметичне елементів матриці
8	3×4	дійсний	Визначити найменший серед парних додатних елементів
9	5×3	цілий	Обчислити суму та кількість парних елементів матриці
10	5×5	дійсний	Обчислити суму та кількість трицифрових елементів
11	4×6	цілий	Обчислити суму та кількість елементів матриці, кратних 3
12	5×4	дійсний	Обчислити модуль суми всіх від'ємних елементів матриці

Закінчення табл. 13.1

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
13	3×5	цілий	Визначити розміщення (індекси) мінімального та максимального елементів матриці
14	4×3	дійсний	Обчислити середнє арифметичне від'ємних елементів
15	6×4	цілий	Обчислити кількість ненульових елементів матриці
16	5×5	дійсний	Визначити максимальний і мінімальний елементи матриці
17	4×5	дійсний	Обчислити середнє арифметичне елементів матриці, значення яких належать проміжку [10, 20]
18	3×5	цілий	Визначити найменший елемент матриці
19	5×3	цілий	Обчислити добуток одноцифрових елементів матриці
20	3×4	дійсний	Обчислити суму модулів всіх від'ємних елементів матриці
21	3×3	цілий	Обчислити суму та кількість двоцифрових елементів
22	5×5	цілий	Обчислити середнє арифметичне елементів, кратних 5
23	4×6	цілий	Визначити найбільший з парних додатних елементів
24	5×4	дійсний	Обчислити суму додатних елементів і кількість від'ємних елементів матриці
25	3×4	цілий	Обчислити суму елементів матриці, значення яких належать проміжку [3, 6]
26	3×3	дійсний	Обчислити визначник матриці
27	6×4	дійсний	Обчислити кількість елементів, що перевищують середнє арифметичне всіх елементів
28	5×5	дійсний	Обчислити середнє арифметичне елементів неголовної діагоналі матриці
29	4×5	цілий	Обчислити суму елементів парних стовпців матриці
30	3×5	дійсний	Визначити мінімальний із додатних елементів матриці

Таблиця 13.2

Варіанти завдань створення функції для змінення елементів матриці

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	4×3	цілий	Замінити парні за значенням (не по індексу) елементи на 0
2	6×4	дійсний	Поміняти місцями мінімальний і максимальний елементи
3	4×4	цілий	Поміняти місцями елементи головної та неголовної діагоналей матриці
4	4×5	дійсний	Замінити всі від'ємні елементи на значення мінімального
5	3×5	цілий	Обчислити суму додатних непарних елементів і замінити кутові елементи матриці на цю суму
6	5×3	цілий	Замінити всі нульові елементи значенням мінімального елемента

Закінчення табл. 13.2

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
7	5×5	цілий	Транспонувати матрицю
8	3×4	дійсний	Замінити нульові елементи на середнє арифметичне найменшого і найбільшого елементів
9	5×3	цілий	Замінити всі непарні елементи матриці одиницями
10	3×5	цілий	Замінити від'ємні елементи матриці нулями
11	4×6	дійсний	Замінити нулями ті елементи матриці, які більші за середнє арифметичне
12	5×5	цілий	Поміняти місцями елементи першого рядка матриці з елементами її неголовної діагоналі
13	4×4	дійсний	Замінити нулями всі елементи від початку і до найбільшого елемента
14	3×5	дійсний	Замінити найменший та найбільший елементи на нулі
15	4×3	цілий	Обчислити суму додатних непарних елементів і замінити парні елементи масиву на цю суму
16	6×4	дійсний	Замінити мінімальний і максимальний елементи значенням середнього арифметичного всіх елементів
17	5×3	дійсний	Замінити парні елементи на значення найменшого елемента
18	4×5	цілий	Замінити парні за значенням елементи матриці на 0
19	3×5	цілий	Поміняти місцями максимальний елемент з першим
20	5×5	дійсний	Розмістити елементи головної діагоналі у зворотному порядку
21	3×4	цілий	Замінити елементи кратні 5 на значення найбільшого елемента
22	3×6	дійсний	Поміняти місцями елементи першого й останнього стовпців
23	5×5	цілий	Замінити на максимальне значення матриці всі його нульові елементи
24	4×6	цілий	Замінити всі парні елементи масиву на значення останнього елемента матриці
25	5×4	дійсний	Поміняти місцями елементи першого й останнього рядків
26	3×4	цілий	Поміняти місцями два найбільші за значенням елементи
27	3×3	дійсний	Замінити елементи головної діагоналі матриці на значення середнього арифметичного її елементів
28	6×4	цілий	Замінити на значення мінімального елемента ті елементи матриці, які менші за середнє арифметичне
29	5×3	дійсний	Замінити від'ємні елементи в непарних рядках матриці на нулі, а парних рядках – на одиниці
30	4×5	цілий	Поміняти місцями два найменші за значенням елементи

Таблиця 13.3

Варіанти завдань створення функції для формування вектора

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
1	3×5	дійсний	Обчислити вектор квадратів елементів мінімальних елементів стовпців матриці
2	5×3	цілий	Обчислити вектор з середньоарифметичних елементів рядків матриці
3	3×4	дійсний	Обчислити вектор як суми елементів стовпців матриці, абсолютне значення яких не перевищує 10
4	6×4	цілий	Обчислити вектор середньоарифметичних додатних елементів рядків матриці
5	4×3	дійсний	Обчислити вектор сум додатних елементів рядків матриці
6	5×4	цілий	Обчислити вектор з середньоарифметичних двоцифрових елементів стовпців матриці
7	4×6	дійсний	Обчислити вектор сум елементів непарних стовпців матриці
8	5×5	цілий	Обчислити вектор з елементів головної діагоналі матриці
9	3×4	дійсний	Обчислити вектор квадратів значень останніх елементів стовпців матриці
10	3×5	цілий	Обчислити вектор добутків непарних елементів стовпців матриці
11	4×4	дійсний	Обчислити вектор сум елементів головної і неголовної діагоналей матриці
12	6×4	цілий	Обчислити вектор з найбільших елементів рядків матриці
13	5×3	дійсний	Обчислити вектор сум перших трьох елементів стовпців матриці
14	4×6	цілий	Обчислити вектор сум непарних елементів стовпців матриці
15	3×5	дійсний	Обчислити вектор середньоарифметичних елементів стовпців матриці
16	4×5	цілий	Обчислити вектор добутків одноцифрових (від 0 до 9) елементів стовпців матриці
17	4×4	дійсний	Обчислити вектор елементів неголовної діагоналі матриці
18	3×6	цілий	Обчислити вектор добутків ненульових елементів стовпців матриці
19	5×5	дійсний	Обчислити вектор сум модулів від’ємних елементів рядків матриці
20	4×6	цілий	Обчислити вектор середньоарифметичних першого і останнього елементів рядків матриці
21	3×4	дійсний	Обчислити вектор з найменших елементів стовпців матриці
22	5×4	цілий	Обчислити вектор середньоарифметичних парних елементів рядків матриці
23	5×3	дійсний	Обчислити вектор сум модулів елементів рядків матриці

Закінчення табл. 13.3

№ вар.	Розмір масиву	Тип даних	Індивідуальне завдання
24	6×4	цілий	Обчислити вектор середньоарифметичних першого й останнього елементів стовпців матриці
25	6×6	дійсний	Обчислити вектор кількостей додатних елементів рядків
26	4×5	цілий	Обчислити вектор сум квадратів елементів стовпців матриці
27	5×3	дійсний	Обчислити вектор квадратів значень останніх елементів рядків матриці
28	6×4	цілий	Обчислити вектор добутків парних елементів непарних рядків матриці
29	5×5	дійсний	Обчислити вектор середньоарифметичних елементів головної і неголовної діагоналей матриці
30	6×5	цілий	Обчислити вектор середньоарифметичних елементів максимального і мінімального елементів стовпців матриці

Лабораторна робота № 14

Створення бібліотеки функцій

Мета роботи: набути практичних навиків зі створення статичної бібліотеки функцій у Visual C++ та долучення її заголовного файла до програмних проектів.

Теоретичні відомості

Статичні та динамічні бібліотеки функцій

Бібліотеками називають набори функцій (підпрограм) і/або об'єктів, звичайно орієнтованих на розв’язання близьких за тематикою завдань. Бібліотеки є хорошим способом повторного використання коду. Замість того щоб щоразу реалізовувати одні і ті самі функції для забезпечення тієї чи іншої функціональності в кожному створюваному проекті, їх можна створити один раз і потім викликати з проектів. Для використання бібліотеки необхідно вказати компілятору, що її потрібно долучити і викликати функцію з бібліотеки (скориставшись відповідним заголовним файлом), при цьому вихідний текст функції не потрібен. З точки зору організації та використання бібліотеки бувають статичними і динамічними.

Статичні бібліотеки (static library) – це набір функцій, але не у формі коду, а у вже скомпільованому вигляді. Файли статичних бібліотек C++ мають розширення `lib`. В результаті зв'язування зі статичною бібліотекою, програма долучає всі використовувані функції, що збільшує її розмір, але робить більш автономною.

Динамічні бібліотеки (dynamic link library – dll) завантажуються операційною системою "на вимогу" запущеної програми вже в ході її виконання. Якщо необхідна бібліотека вже була завантажена в пам'ять, то повторне завантаження не виконується. При цьому один і той самий набір функцій чи об'єктів бібліотеки може бути використаний одночасно кількома працюючими програмами, що дозволяє ефективно використовувати ресурси оперативної пам'яті. Динамічні бібліотеки в Windows звичайно мають розширення `dll`.

Заголовні файли

Заголовні файли мають розширення `h` і містять заголовки (прототипи) функцій, а також оголошення типів, констант і змінних з ключовим словом `extern`. Перевага заголовних файлів полягає у тому, що, коли створено заголовний файл для одного модуля (одного або декількох `cpp`-файлів) програми, можна переносити всі зроблені оголошення до іншої програми C++. Для цього слід просто залучити заголовний файл за допомогою директиви `#include`. Зазвичай для кожного `cpp`-файла створюється власний заголовний файл з таким самим ім’ям і розширенням `h`. І навпаки, здебільшого для кожного заголовного файла створюється `cpp`-файл з реалізацією функцій, оголошених у `h`-файлі.

Заголовні файли можна розділити на стандартні й створювані програмістом. Заголовні файли, створені програмістом, звичайно розміщують у теці проекту. Імена цих файлів у директиві `#include` пишуться у подвійних лапках і завжди з розширенням `h`, наприклад: `#include "MyBibl.h"`.

Директиви препроцесора

Директиви препроцесора є інструкціями, записаними в тексті С-програми, які виконуються до трансляції програми. Директиви препроцесора дозволяють вставляти програмний код з іншого файла, забороняти трансляцію частини тексту тощо. Всі директиви препроцесора розпочинаються зі знаку #. Після директив препроцесора крапка з комою не ставиться.

Найбільш поширені препроцесорні директиви:

`#include` – долучення до програми вмісту зазначеного файла;

`#define` – визначення макросу або препроцесорного ідентифікатора;

`#pragma` – дії, передбачені реалізацією;

`#ifdef` – перевірка визначеності ідентифікатора;

`#ifndef` – перевірка невизначеності ідентифікатора;

`#error` – формування тексту повідомлення про помилку при трансляції;

Наприклад, щоб задати константу `n` зі значенням 10 слід написати директиву

```
#define n 10
```

Ця директива є аналогом такої інструкції `const int n = 10;`

Відмінності полягають у тому, що:

1) при використанні константи пам'ять під константну змінну виділяється, а при використанні директиви – ні;

2) при використанні директиви не виконується перевірка типу значення, що може спричинити помилки у програмі;

3) тип значення `n` при використанні директиви визначається автоматично, тобто, якщо у програмі є суттєвим, щоб `n` мала тип `unsigned short`, зробити це за допомогою директиви неможливо.

Отже, використання директиви `#define` у C++ без особливої на те потреби є небажаним.

Директива `#include` долучає до тексту програми вміст зазначеного файла. Ця директива широко використовується для долучення до програм заголовних файлів бібліотек. При чому імена заголовних файлів стандартних бібліотек зазвичай записують у кутових дужках, а власних бібліотек – у лапках, наприклад:

```
#include "MyBibl.h"
#include <math.h>
```

Дія директиви `#pragma` залежить від конкретної реалізації компілятора. Ця директива дозволяє видавати компілятору різні інструкції.

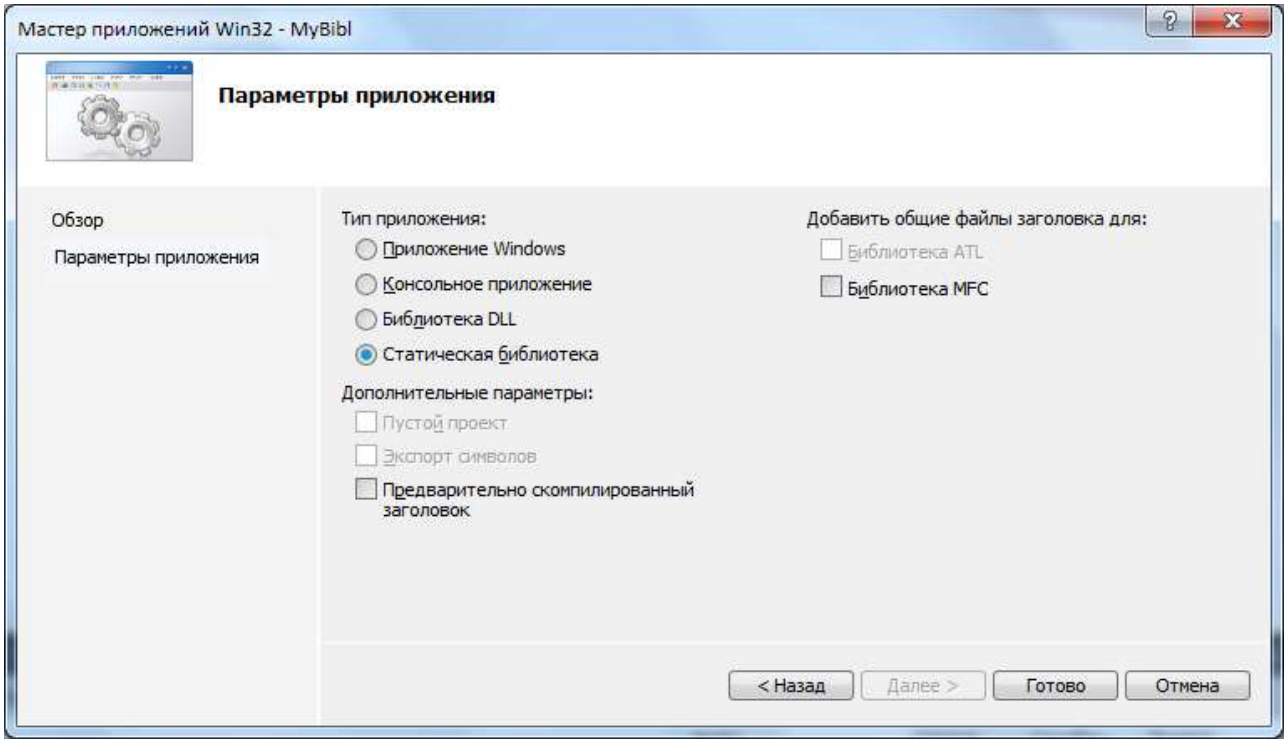
Директива `#if` та її модифікації `#ifdef`, `#ifndef` разом з директивами `#else`, `#endif`, `#elif` дозволяють організувати умовне опрацювання тексту програми. При використанні цих засобів компілюється не весь текст, а лише ті його частини, які вибираються за допомогою вищенаведених директив. Головна ідея полягає у тому, що, якщо вираз, який розміщено після директив `#if`, `#ifdef`, `#ifndef` виявиться істинним, то буде скомпільовано код, розміщений між однією з цих трьох директив та директивою `#endif`, інакше цей код буде опущений. Директива `#endif` використовується для позначення закінчення блока `#if`. Директиву `#else` можна використовувати з кожною із наведених вище директив для надання альтернативного варіанта компіляції.

Директива `#error` дозволяє задавати текст діагностичного повідомлення, яке виводиться при виявленні помилок. За наявності цієї директиви відображується задане повідомлення і номер рядка.

Послідовність створення статичної бібліотеки з LIB-файлом

1) Для створення проекту статичної бібліотеки слід виконати команду *Файл / Создать / Проект* (або натиснути клавіші *Ctrl+Shift+N*). У лівій частині діалогового вікна розгорнути *Установленные шаблоны / Visual C++ / Win32*, а в центральній частині вікна вибрати *Консольное приложение Win32*. У цьому ж вікні записати ім'я проекту, наприклад MyBibl, за допомогою кнопки *Обзор* вибрати необхідну теку для зберігання проекту і натиснути кнопку *ОК*.

2) У діалоговому вікні *Мастер приложений Win32* натиснути кнопку *Далее*, а тоді вибрати для поля *Тип приложения* значення *Статическая библиотека* і для поля *Дополнительные параметры* вимкнути прапорець *Предкомпилированный заголовок*. Натиснути кнопку *Готово*.



3) Для створення файла заголовка відкрити контекстне меню на імені проекту MyBibl, а тоді в полі *Обозреватель решений* вибрати *Добавить / Создать элемент*. У діалоговому вікні *Добавление нового элемента* в лівій частині у розділі *Visual C++* вибрати *Код*, а центральній частині – *Заголовочный файл (.h)*. Вказати ім'я для заголовка файла, наприклад MyBibl.h, і натиснути кнопку *Добавить*. Після цього з'явиться вікно порожнього заголовного файла, в якому слід вписати прототипи функцій та інші необхідні оголошення, наприклад такі:

```
#ifndef MyBiblH
#define MyBiblH
#define m 5
#define n 8
const double Pi = 3.14159;
```

```
// Оголошення типів matr (“матриця”) і vect (“вектор”)
typedef long double matr[m][n],  vect[m];
// Оголошення прототипів функцій
void elem_Matr(matr a);
void elem_Vect(matr a, vect v);
int G(matr a);
#endif
```

4) Для створення файла реалізації слід відкрити контекстне меню на імені проекту MyBibl і в полі *Обозреватель решений* вибрати *Добавить / Создать элемент*. У діалоговому вікні *Добавление нового элемента* в лівій частині вікна у розділі Visual C++ вибрати *Код*, а в центральній частині вибрати *Файл C++ (.cpp)*. Вказати ім’я файла, наприклад MyBibl.cpp, і натиснути кнопку *Добавить*. Після цього з’явиться вікно порожнього cpp-файла, в якому слід вписати програмний код реалізації оголошених у MyBibl.h функцій:

```
#include "MyBibl.h"
#include <math.h>     // Долучення математичної бібліотеки
// Визначення функції обчислення елементів матриці
void elem_Matr( matr a)
{ for (int i=0; i<m; i++)
  for (int j=0; j<n; j++)
    a[i][j] = tan(exp((i+4.)/(j+2)))+log10(i+j+1./3);
}
// Визначення функції обчислення елементів вектора
// як стовпця з максимальною сумою елементів
 void elem_Vect(matr a, vect v)
 { double x[n];
   for (int j=0; j<n; j++)
   { x[j]=0;
     for (int i=0; i<m; i++) x[j]+=a[i][j];
   }
   int ind=0;
   for (int j=0; j<n; j++)
     if (x[j]>x[ind]) ind=j;
   for (int i=0; i<m; i++) v[i]=a[i][ind];
}
// Обчислення кількості додатних елементів матриці
 int G (matr a)
 { int kol=0, i, j;
   for (i=0; i<m; i++)
   for (j=0; j<n; j++)
     if (a[i][j]>0) kol++;
   return kol ;
 }
```

5) Для компіляції статичної бібліотеки слід виконати команду *Построение / Построить решение* (або натиснути [F7]). В результаті буде створена статична бібліотека MyBibl.lib, яку можна використовувати іншими програмами.

6) Щоб створити консольний програмний проект LbBibl, в якому будуть використовуватись функції з бібліотеки MyBibl.lib слід виконати такі дії:

– команду *Файл / Создать / Проект / Консольное приложение Win32*;

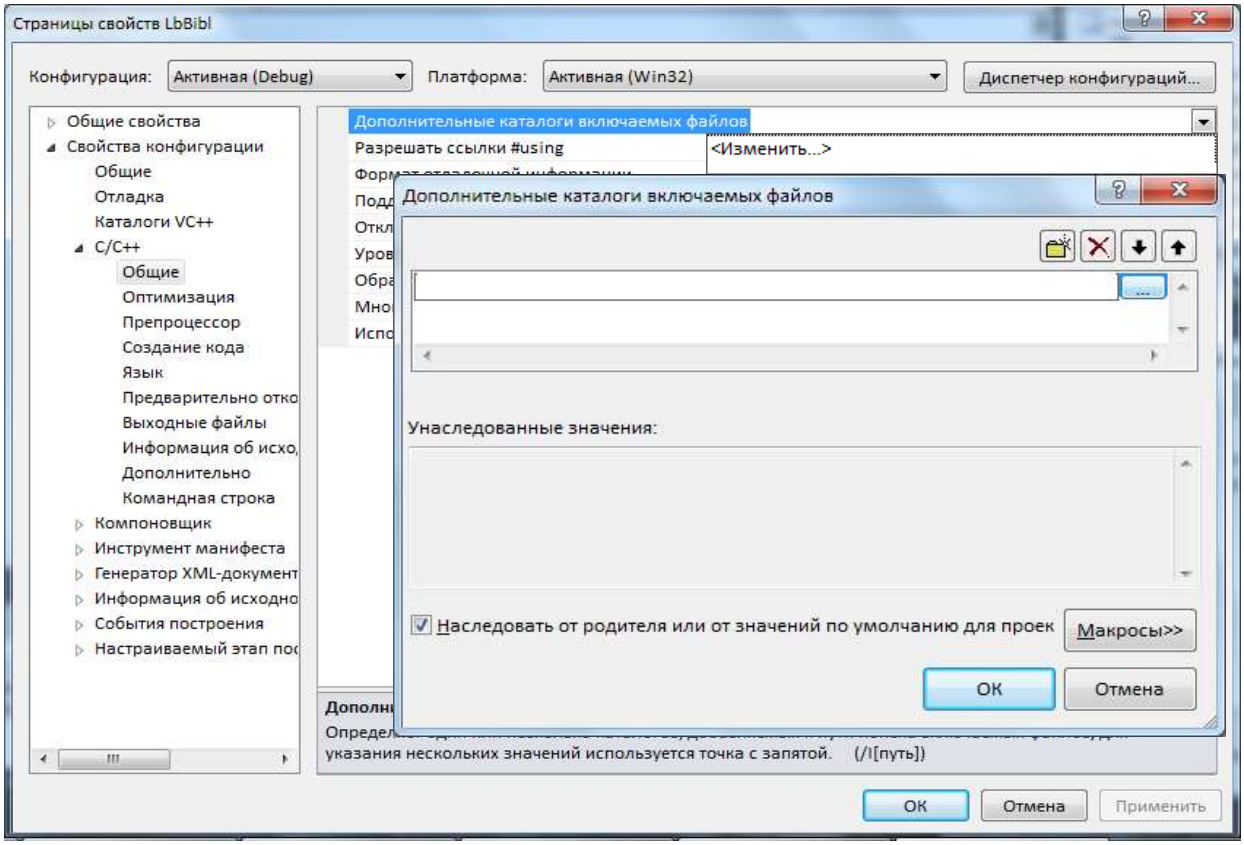
– вписати ім’я проекту, наприклад LbBibl. За потреби вибору власної теки для зберігання проекту можна скористатись кнопкою *Обзор*. Натиснути *ОК*;

– у діалоговому вікні *Мастер приложений Win32*, що з’явиться, спочатку натиснути кнопку *Далее*, а тоді вказати для поля *Тип приложения* пункт *Консольное приложение*, а для поля *Дополнительные параметры* зняти позначку *Предкомпилированный заголовок*. Натиснути *Готово*.

Після цього з’явиться вікно cpp-файла проекту, для якого слід створити посилання на розміщення файлів статичної бібліотеки MyBibl (див. пп. 8 … 11).

7) Для зручності доцільно скопіювати три необхідні файли (MyBibl.lib, MyBibl.h та MyBibl.cpp) до теки основного проекту LbBibl/LbBibl. Саме так долучаються до створюваних проектів бібліотеки створені раніш. Слід зазначити, що до цього двоє з цих файлів були розміщені в теці MyBibl/MyBibl, а один файл – в MyBibl/Debug. І, якщо не виконати таке копіювання, то в наступному п. 8 слід буде вибрати теку MyBibl/MyBibl, а в п. 9 – теку MyBibl/Debug. Але при цьому слід будь-які змінення у файлах бібліотеки здійснювати не в скопійованих файлах, а в "оригіналі" бібліотеки з обов’язковою перекомпіляцією і перекопіюванням зазначених трьох файлів.

8) Для зазначення місця розміщення бібліотеки MyBibl слід у віконці *Обозреватель решений* (якщо воно закрите, натиснути *Ctrl+w,s*) на імені проекту LbBibl натиснути праву кнопку миші і вибрати з контекстного меню *Свойства*.



У діалоговому вікні *Страницы свойств LbBibl* вибрати *Свойства конфигурации / C/C++ / Общие / Дополнительные каталоги включаемых файлов*, роз-

крити список (кнопка), натиснути кнопку ... і вибрати теку із заголовним файлом бібліотеки (якщо було виконано копіювання файлів у п. 7 – теку LbBibl/LbBibl, а якщо ні – теку MyBibl/MyBibl).

9) У цьому самому вікні перейти з категорії *C/C++* до категорії *Компоновщик / Общие / Дополнительные каталоги библиотек*, розкрити список (кнопка), натиснути кнопку ... і вибрати теку із файлом бібліотеки MyBibl.lib (якщо у п. 7 файли було скопійовано – теку LbBibl/LbBibl, а якщо ні – MyBibl/Debug).

10) Перейти до категорії *Компоновщик / Ввод / Дополнительные зависимости*, розкрити список (кнопка) і вписати ім'я файла бібліотеки – MyBibl.lib. Натиснути кнопки *Применить* і *ОК*.

11) Якщо після вказаних кроків у вікні *Обозреватель решений* у розділі *Заголовочные файлы* не з'явився файл MyBibl.h та у розділі *Файлы исходного кода* – файл MyBibl.cpp, то їх можна долучити в такий спосіб: натиснути правою кнопкою миші на *Заголовочные файлы* та виконати команду контекстного меню *Добавить / Существующий элемент*, вибравши файл MyBibl.h. Аналогічно можна долучити файл MyBibl.cpp до розділу *Файлы исходного кода*.

12) Тепер функції з бібліотеки MyBibl можна викликати в програмі проекту LbBibl. Для цього змінити вміст файла LbBibl.cpp на такий код:

```
#include "MyBibl.h"
#include <iostream>
#include <iomanip>  // для використання маніпулятора setprecision
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{ setlocale(0,".1251");
  int i,j;  matr a;   vect x;
  elem_Matr(a);
  cout<<"\nМатриця:"<<endl;
  for (i=0; i<m; i++)
  { for (j=0; j<n; j++) cout<< fixed << setprecision(2) <<a[i][j]<<"\t";
    cout << endl;
  }
  elem_Vect(a, x);
  cout<<"\nВектор (стовпець з максимальною сумою елементів):"<<endl;
  for (i=0; i<m; i++) cout<< fixed << setprecision(2)  <<x[i]<<"\n";
  cout<<"\nКількість додатних елементів матриці: "<<G(a)<<endl;
  system("pause>>void");
  return 0;
}
```

Результати роботи:

```
Матриця:
1.52    0.89    -0.08   -0.78   -1.89   -4.21   -12.01  90.36
-0.28   -1.15   0.89    0.19    -0.39   -1.16   -2.40   -4.83
3.24    2.52    4.89    0.91    0.35    -0.13   -0.72   -1.56
-7.20   1.87    0.14    2.10    0.94    0.47    0.05    -0.43
3.14    -3.11   2.80    -3.21   1.68    0.96    0.56    0.20

Вектор (стовпець з максимальною сумою елементів):
90.36
-4.83
-1.56
-0.43
0.20

Кількість додатних елементів матриці: 22
```

Приклад створення бібліотеки функцій і використання її функцій в основній програмі

Завдання. Створити бібліотеку з трьома функціями для реалізації таких задач:

1) обчислення елементів матриці a розмірністю 8×8 за формулою

$$a_{i,j}=\begin{cases}\sqrt[3]{\left(\dfrac{i+j+2}{7i+1}+j\right)^4}-3.7(i+j), & \text{за} \quad \sin(i+j)>0;\\ \pi\sin^3(i-j), & \text{за} \quad \sin(i+j)\le 0;\end{cases}$$

2) обчислення вектора (одновимірного масиву), елементи якого є сумами елементів відповідних рядків матриці a;

3) пошук елемента матриці a, найбільш наближеного до значення середнього арифметичного елементів матриці a.

Розв'язок. Послідовність створення статичної бібліотеки з MyBibl.lib дивись у відповідному пункті теоретичних відомостей цієї лабораторної роботи.

У файлі MyBibl.h слід написати потрібні оголошення й заголовки функцій.

```
#ifndef MyBiblH
#define MyBiblH
#define k 8
#define Pi 3.1415
// Оголошення типів matr (“матриця”) і vekt (“вектор”)
typedef long double matr[k][k],  vekt[k];
// Оголошення прототипів функцій
void elem_Matr( matr c);
void elem_Vect(matr c, vekt v);
double G(matr c);
#endif
```

У файлі MyBibl.cpp прописати визначення функцій, прототипи яких розміщено у файлі MyBibl.h.

```
#include "MyBibl.h"
#include <math.h>     // Долучення математичної бібліотеки
// Визначення функції обчислення елементів матриці
void elem_Matr( matr c)
{ for(int i=0; i<k; i++)
  for(int j=0; j<k; j++)
    if(sin(double(i+j))>0) c[i][j] = pow((i+j+2.)/(7*i+1)+j,4./3)-3.7*(i+j);
    else  c[i][j] = Pi*pow( sin(double(i-j)),3);
}
// Визначення функції обчислення елементів вектора
 void elem_Vect(matr c, vekt v)
 { for(int i=0; i<k; i++)
   { v[i]=0;
     for(int j=0; j<k; j++) v[i]+=c[i][j];
 } }
```

```
// Пошук найближчого елемента до середнього арифметичного
 double G (matr c)
 { int ni=0, nj=0, i, j;     double sr=0;
   for (i=0; i<k; i++)
   for (j=0; j<k; j++) sr += c[i][j]/pow(k,2.0);

   for (i=0; i<k; i++)
   for (j=0; j<k; j++)
     if (fabs(sr-c[i][j]) < fabs(sr-c[ni][nj]))
      {  ni=i;     nj=j; }
   return c[ni][nj] ;
 }
```

Головний файл проекту:

```
#include "MyBibl.h"
#include <iostream>
#include <clocale>
#include <iomanip>  // для використання маніпулятора setprecision
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{ setlocale(0,".1251");
  int i,j;
  matr a;   vekt x;
  elem_Matr( a);
  cout<<"\nМатриця:"<<endl;
  for (i=0; i<k; i++)
  { for (j=0; j<k; j++) cout<< fixed << setprecision(2) <<a[i][j]<<"\t";
    cout << endl;
  }
  elem_Vect(a, x);
  cout<<"\nВектор:"<<endl;
  for (i=0; i<k; i++) cout<< fixed << setprecision(2)  <<x[i]<<"\t";
  cout << endl;
  cout<<"\nСкалярне значення: "<<G(a)<<endl;
  system("pause>>void");
  return 0;
}
```

Результати роботи консольного додатка:

```
Матриця:
0.00    2.65    3.50    4.90    1.36    2.77    0.07    14.42
-3.43   -5.68   -7.48   -2.36   -0.01   1.36    -12.19  -12.93
-7.23   -9.63   0.00    -1.87   -2.36   -15.96  -17.05  -18.01
-10.96  2.36    1.87    0.00    -18.67  -20.00  -21.17  1.36
-1.36   0.01    2.36    -20.97  -22.51  -23.87  -2.36   -0.01
-2.77   -1.36   -22.95  -24.73  -26.30  0.00    -1.87   -2.36
-0.07   -24.61  -26.68  -28.47  2.36    1.87    0.00    -33.81
-25.80  -28.32  -30.40  -1.36   0.01    2.36    -36.46  -37.59

Вектор:
29.67   -42.72  -72.11  -65.20  -68.71  -82.34  -109.42 -157.57

Скалярне значення: -9.63
```

Питання та завдання для самоконтролю

1) Що являють собою бібліотеки функцій C++?
2) Яке призначення заголовних файлів?
3) Що записують у файлах реалізації?
4) Для чого у мові C++ використовують директиви препроцесора? Назвіть відомі Вам директиви.
5) За допомогою якої директиви долучають заголовні файли?

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою C++ програмний код заголовного файла і файла реалізації бібліотеки з трьома функціями для розв'язання завдань, поданих в табл. 14.1 відповідно до індивідуального варіанта. Написати програмний код основної програми з долученням написаної бібліотеки, викликом її функцій і виведенням результатів обчислень.

3) Створити на комп'ютері в середовищі Visual C++ власну статичну бібліотеку з розробленими у п. 2 трьома функціями та програмний проект з її долученням. Занести результати обчислень до протоколу.

Таблиця 14.1

Варіанти завдань для створення бібліотеки

№ вар.	Індивідуальне завдання
1	1) Обчислити елементи матриці розміром 8×3 за формулою: $a_{ij} = \left(\sin^2 i + \cos^2 j\right)^{\frac{i-5}{j+1}} + 7{,}45\,\mathrm{tg}\left(\frac{i-5}{j+1.5}\right)$ 2) Обчислити елементи вектора як суми елементів стовпців матриці. 3) Значення різниці між абсолютними значеннями максимального та мінімального елементів матриці
2	1) Обчислити елементи матриці розміром 4×6 за формулою: $a_{ij} = \ln^3(7.3i - j + 8) - \frac{\mathrm{e}^j}{j+1}$ 2) Обчислити елементи вектора як середнє арифметичне рядків матриці. 3) Значення суми додатних елементів матриці
3	1) Обчислити елементи матриці розміром 6×6 за формулою: $a_{ij} = \left(-\frac{2+i}{3+j}\right)^i - \mathrm{e}^{\cos j}$ 2) Обчислити елементи вектора як максимальні елементів рядків матриці. 3) Значення середнього арифметичного елементів матриці з парними індексами

Продовження табл. 14.1

№ вар.	Індивідуальне завдання
4	1) Обчислити елементи матриці розміром 7×7 за формулою: $a_{ij} = \mathrm{e}^{\frac{i+j}{12.5}} - \sqrt{\lvert \mathrm{ctg}(i+j-50.5) \rvert}$ 2) Обчислити елементи вектора як суми елементів головної та неголовної діагоналей матриці. 3) Значення добутку ненульових елементів матриці
5	1) Обчислити елементи матриці розміром 5×7 за формулою: $a_{ij} = \left(\frac{2}{3j+1.5}\right)^{i} - \lg\frac{\mathrm{e}^{i}}{3j+9}$ 2) Обчислити елементи вектора як стовпець матриці з максимальним елементом. 3) Значення добутку елементів неголовної діагоналі матриці
6	1) Обчислити елементи матриці розміром 7×6 за формулою: $a_{ij} = \left(\frac{2+i}{3i+j+1}\right)^{i+j} + \cos\left(\mathrm{e}^{\cos(ij-1.5)}\right)$ 2) Обчислити елементи вектора як середнє арифметичне стовпців матриці. 3) Значення суми модулів від'ємних елементів матриці
7	1) Обчислити елементи матриці розміром 8×7 за формулою: $a_{ij} = \left(\frac{\cos^2(i+1.5)}{2.5+j}\right)^{\sin(i+j+1.5)} + \mathrm{e}^{\frac{i}{j+5.5}}$ 2) Обчислити елементи вектора як рядок матриці з мінімальним елементом. 3) Значення кількості додатних елементів матриці, менших 5-ти
8	1) Обчислити елементи матриці розміром 7×7 за формулою: $a_{ij} = \lg^4(i+j+1.2) - 0.5^{i}$ 2) Обчислити елементи вектора як суми елементів рядків матриці з парними індексами стовпців. 3) Значення середньоарифметичного елементів головної діагоналі матриці
9	1) Обчислити елементи матриці розміром 8×5 за формулою: $a_{ij} = (\sin i \cdot \ln(j+1.2))^3 - \frac{(-1.3)^{i+j}}{\lg(i+j+0.2)}$ 2) Обчислити елементи вектора як середнє арифметичне додатних елементів рядків матриці. 3) Значення мінімального додатного елемента матриці
10	1) Обчислити елементи матриці розміром 8×5 за формулою: $a_{ij} = \cos(2i+j)\lg(i+5j+0.3)$ 2) Обчислити вектор як суми від'ємних елементів стовпців матриці. 3) Значення кількості елементів матриці, більших за 1

Продовження табл. 14.1

№ вар.	Індивідуальне завдання
11	1) Обчислити елементи матриці розміром 5×5 за формулою: $a_{ij} = \operatorname{ctg}(3i + 2j + 5) + \ln(i + 2j + 2)$ 2) Обчислити елементи вектора як квадрати елементів неголовної діагоналі матриці. 3) Сума найменшого та найбільшого елементів матриці
12	1) Обчислити елементи матриці розміром 4×7 за формулою: $a_{ij} = \ln \frac{i + 2j + 2}{e^{2i-j}} + (0.5i)^{\frac{1}{j+1}}$ 2) Обчислити елементи вектора як рядок матриці з максимальним елементом. 3) Значення середнього арифметичного елементів другого рядка матриці
13	1) Обчислити елементи матриці розміром 6×5 за формулою: $a_{ij} = \sqrt[3]{\lg\left(i^2 + j^2 + 2.5\right)}(i - 2.5)^j$ 2) Обчислити елементи вектора як середнє арифметичне елементів другого та останнього стовпців матриці. 3) Значення добутку елементів матриці зі значенням меншим за 10
14	1) Обчислити елементи матриці розміром 8×4 за формулою: $a_{ij} = \begin{cases} \operatorname{ctg}(i - j + 7) + \lg\left(\frac{2i - j + 7}{i^2 - j^2 + 11.2}\right), & \text{при } j \neq i \\ \ln^4(i + j + 2.3), & \text{при } j = i \end{cases}$ 2) Обчислити елементи вектора як різниці елементів першого та третього стовпців матриці. 3) Значення кількості елементів зі значенням більшим за середнє арифметичне елементів матриці
15	1) Обчислити елементи матриці розміром 6×6 за формулою: $a_{ij} = \sin(i + j - 7.2)^3 \ln(9.2i - j + 7.2)$ 2) Обчислити елементи вектора як скалярний добуток елементів рядків матриці на останній стовпець. 3) Значення максимального елемента головної діагоналі матриці.
16	1) Обчислити елементи матриці розміром 8×6 за формулою: $a_{ij} = (-1.3)^{i+j} \ln\left(3^{i-j+6.2}\right)$ 2) Обчислити елементи вектора як середнє арифметичне елементів першого та останнього рядків матриці. 3) Значення максимального від'ємного елемента матриці
17	1) Обчислити елементи матриці розміром 5×7 за формулою: $a_{ij} = \tan(j - 4i + 0.5) + \lg(3i + 2j + 2)$ 2) Обчислити вектор як рядок з мінімальною сумою елементів матриці. 3) Значення кількості елементів матриці, абсолютне значення яких менше 1

Продовження табл. 14.1

№ вар.	Індивідуальне завдання
18	1) Обчислити елементи матриці розміром 5×8 за формулою: $a_{ij} = \ln^3(9i + j + 1) - \mathrm{e}^i$ 2) Обчислити елементи вектора як стовпець матриці з мінімальною сумою елементів. 3) Сума елементів матриці
19	1) Обчислити елементи матриці розміром 8×7 за формулою: $a_{ij} = \left(\sin(7 + i)\right)^{j-7} + \mathrm{e}^{\cos(i+j)}$ 2) Обчислити елементи вектора як суми елементів стовпців матриці, значення яких більше за 10. 3) Значення середньоарифметичного елементів останнього рядка матриці
20	1) Обчислити елементи матриці розміром 8×8 за формулою: $a_{ij} = \lg\left(\mathrm{e}^{\frac{7+i}{j+2}}\right) - \sqrt{\lvert\mathrm{tg}(j+1)\rvert}$ 2) Обчислити елементи вектора як скалярний добуток рядків матриці на другий стовпець. 3) Значення добутку елементів останнього стовпця матриці
21	1) Обчислити елементи матриці розміром 8×5 за формулою: $a_{ij} = \begin{cases} \lg(i-j) + \cos(7i - j), & \text{при } i > j \\ \ln(j - i + 7) - \mathrm{e}^{\frac{i}{j-i+1}}, & \text{при } i \le j \end{cases}$ 2) Обчислити елементи вектора як найбільші за модулем елементи рядків. 3) Значення суми елементів матриці зі значенням більшим за середнє арифметичне елементів
22	1) Обчислити елементи матриці розміром 7×7 за формулою: $a_{ij} = \left(\frac{3}{9j - i}\right)^i - \sin\left(\mathrm{e}^{ij}\right)$ 2) Обчислити елементи вектора як скалярний добуток елементів стовпців матриці на перший рядок. 3) Значення кількості від'ємних елементів у перших трьох рядках матриці
23	1) Обчислити елементи матриці розміром 8×5 за формулою: $a_{ij} = \ln^{2.5}(4i + j + 1.8)\cos(ij)$ 2) Обчислити елементи вектора як стовпець з мінімальною сумою елементів. 3) Значення середнього арифметичного елементів четвертого рядка матриці.
24	1) Обчислити елементи матриці розміром 8×7 за формулою: $a_{ij} = \sqrt{\lvert i - 0.3j\rvert}(\cos i + \sin j)^2 - \ln 2$ 2) Обчислити вектор як мінімальні за модулем елементи рядків матриці. 3) Значення кількості додатних елементів матриці

Закінчення табл. 14.1

№ вар.	Індивідуальне завдання
25	1) Обчислити елементи матриці розміром 3×6 за формулою: $a_{ij} = (\sin(i-1) - \cos j)^3 + \ln\frac{i+7}{j+4}$ 2) Обчислити елементи вектора як квадрати значень останніх елементів стовпців матриці. 3) Модуль суми всіх від'ємних елементів матриці
26	1) Обчислити елементи матриці розміром 8×3 за формулою: $a_{ij} = \mathrm{tg}\left(\mathrm{e}^{\frac{i+4}{j+2}}\right) - \frac{-6^{i+j-1}}{\mathrm{e}^{3i-2j}}$ 2) Обчислити елементи вектора як кількості додатних елементів рядків. 3) Середнє арифметичне елементів матриці, значення яких належать діапазону [10, 20]
27	1) Обчислити елементи матриці розміром 8×4 за формулою: $a_{ij} = \sqrt{\lg\left(\frac{i+7}{j+4}\right)} - (-2)^{i+j-1}$ 2) Обчислити вектор як модуль суми елементів рядків матриці. 3) Значення кількості від'ємних елементів матриці
28	1) Обчислити елементи матриці розміром 8×8 за формулою: $a_{ij} = \frac{\sqrt{\lvert\ln(i+2j+2.4)\rvert}}{\sin(i+j+0.5)}$ 2) Обчислити елементи вектора як скалярний добуток елементів рядків матриці на головну діагональ. 3) Значення середнє арифметичне елементів неголовної діагоналі матриці.
29	1) Обчислити елементи матриці розміром 6×6 за формулою: $a_{ij} = \frac{2.1^{i+j}}{\ln(i+2)} - (-2)^{i+j}$ 2) Обчислити елементи вектора як середньоарифметичне максимального і мінімального елементів стовпців матриці. 3) Значення суми елементів головної діагоналі матриці
30	1) Обчислити елементи матриці розміром 8×6 за формулою: $a_{ij} = \begin{cases} \mathrm{e}^{\frac{i}{j+2}}\sin(i-j), & \text{при } i \neq j \\ \arcsin((i+j-2)/15), & \text{при } i = j \end{cases}$ 2) Обчислити елементи вектора як суми максимального і мінімального елементів рядків матриці. 3) Значення найменшого елемента матриці

Лабораторна робота № 15

Вказівники і динамічна пам'ять при опрацюванні одновимірних масивів

Мета роботи: набути практичних навиків програмного використання вказівників та динамічної пам’яті при опрацюванні одновимірних масивів.

Теоретичні відомості

Вказівники

Вказівник – це змінна, яка містить адресу іншої змінної, при чому вказівник вказує на змінну того типу, адресу якої він містить.

Вказівник оголошується за допомогою *:

<тип> `*`*<ім’я>*`;`

Тут *тип* – це базовий тип вказівника, котрим може бути який завгодно тип. *Ім’я* є ідентифікатором змінної-вказівника. Слід звернути увагу на те, що *тип* – це не тип вказівника, а тип даних, адресу яких буде записано у вказівнику.

Наприклад, оголошення вказівників `A` та `B` на ціле число та дійсне числа відповідно:

```
int *A;  double *B;
```

У мові C++ визначено дві операції для роботи з вказівниками:

1) `&` – **адресація**, тобто отримання адреси змінної. Приміром, щоб отримати адресу змінної `x`, оголошеної як `int x=10`, можна скористатись командою `A = &x;`

2) * – **розадресація** чи розіменування вказівника. Щоб взяти значення 10, з адреси у вказівнику `A` слід звернутися `*A`, а щоб змінити це значення, збільшивши його на 1, можна скористатись командою `(*A)++` (при чому команда `*A++` дасть помилкове рішення).

Звернімо увагу на те, що "зірочки" при оголошенні вказівника та операції розадресації – це зовсім різні речі, які лише позначаються однаковим символом.

Ініціалізувати вказівник треба до його використання, тобто присвоїти йому адресу якоїсь змінної або об’єкта. Вказівник, який не вказує на жодне значення, називається порожнім, чи нульовим. Такий вказівник має значення 0 чи NULL.

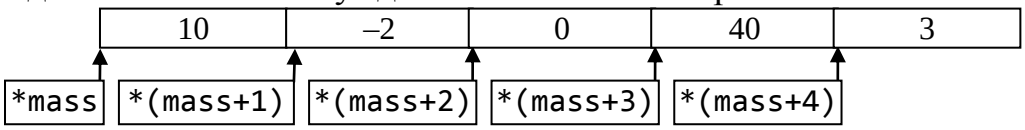
Вказівники на одновимірні масиви

Масиви та вказівники у C++ тісно пов’язані і можуть використовуватись майже еквівалентно. Ім’я масиву можна сприймати як константний вказівник на перший елемент масиву. Його відмінність від звичайного вказівника полягає у тому, що його неможна модифікувати.

Здійснимо оголошення масиву `mass` з п’яти цілих чисел з ініціалізацією значень елементів і вказівника на ціле `ptr`:

```
int mass[5] = {10,-2,0,40,3}, *ptr;
```

При такому оголошенні масиву пам’ять виділиться не лише для п’яти елементів масиву, а й для вказівника з ім’ям `mass`, значення якого дорівнюватиме адресі першого елемента масиву `mass[0]`, тобто сам масив залишиться безіменним, а доступ до елементів масиву здійснюватиметься через вказівник з ім’ям `mass`.



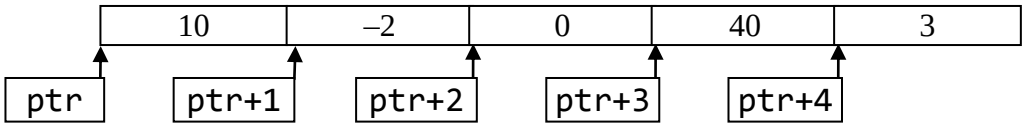
Для того, щоб звернутися до 3-го елемента цього масиву, можна записати чи то `mass[2]` чи `*(mass+2)`. При реалізації на комп’ютері перший з цих способів зводиться до другого, тобто індексний вираз перетвориться до адресного. Оскільки операції над вказівниками виконуються швидше, тому, якщо елементи масиву вибробляються почергово, то доцільніше використовувати другий спосіб. Якщо ж вибір елементів є випадковим, то, щоб уникнути помилок, прийнятнішим є перший спосіб. Крім того, перший спосіб є більш наочним і звичним для сприйняття, що сприяє кращій читабельності програм.

Оскільки ім’я масиву є вказівником на перший елемент масиву, можна надати вказівнику адресу першого елемента масиву за допомогою оператора:

```
ptr = mass;
```

Цей запис є еквівалентним присвоюванню адреси першого елемента масиву (тобто до елемента з нульовим індексом) у вигляді оператора

```
ptr = &mass[0];
```



Після цього звернутися до першого елемента масиву і надати йому значення 2 можна будь-яким з шести операторів:

```
* mass = 2;     mass[0] = 2;       *(mass+0) = 2;
* ptr = 2;      ptr[0] = 2;        *(ptr+0) = 2;
```

Усі ці оператори за дією є тотожними, але швидше за всі виконуватимуться присвоювання `*mass = 2` та `*ptr = 2`, оскільки в них не треба виконувати операції додавання.

Вказівники можна індексувати так само, як і масиви. Наприклад, вираз `ptr[3]` посилається до четвертого елемента масиву `mass[3]`.

Зауважимо, що не слід плутати такі оголошення:

```
int *p1[10];                    // приклад 1
int (*p2)[10];                  // приклад 2
```

У першому прикладі оголошено масив вказівників з ім’ям `p1`. Масив складається з 10-ти елементів, кожний з яких є вказівником на змінну типу `int`.

У другому прикладі оголошено змінну-вказівник з ім’ям `p2`, яка вказує на масив з 10-ти цілих чисел типу `int`.

Динамічна пам'ять

Окрім звичайної пам’яті (стека), в якій автоматично розміщуються змінні за їхнього оголошення, існує ще й *динамічна пам’ять* (heap − купа), в якій змінні можуть розміщуватися динамічно. Це означає, що пам’ять виділяється під час виконання програми, й лише тоді, коли у програмі зустрінеться спеціальна інструкція. Основна потреба у динамічному виділенні пам’яті виникає, коли розмір або кількість даних заздалегідь є невідомі, а визначаються в процесі виконання програми.

У C++ існує кілька команд для виділення динамічної пам’яті.

1) Найчастіше для виділення динамічної пам'яті використовується оператор **new**, який у загальному вигляді записується як:

*<тип> *<ім’я_вказівника> =* **new** *<тип>*;

Наприклад, оператор створення *динамічного масиву* з 10-ти елементів:

```
int *a = new int [10];
```

при цьому виділяється пам’ять для 10-ти цілих чисел. Звернутися до кожного з цих чисел можна за його номером: `a[0]`, `a[1]` і т. д. або через вказівник: `*a` − те ж саме, що `a[0]`; `*(a+1)` − те ж саме, що `a[1]` і т. д.

Для звільнення динамічної пам’яті, виділеної за допомогою оператора `new`, використовується оператор **`delete`**:

`delete` *<вказівник>*;

Наприклад:

```
delete []a;
```

Крім операторів `new` та `delete`, існують функції, які перейшли до C++ з C, але вони використовуються на практиці набагато рідше.

2) Виділення динамічної пам'яті функцією `malloc`, яка має такий формат:

```
void *malloc(size_t size);
```

Єдиний аргумент цієї функции `size` − кількість байтів, яку треба виділити. Функція повертає вказівник на початок виділеної пам’яті. Якщо для розміщення заданої кількості байтів є недостатньо пам’яті, функція `malloc()` повертає `NULL`. Вміст ділянки лишається незмінним, тобто там може залишитися “бруд”. Якщо аргумент `size` дорівнює 0, функція повертає `NULL`. Наприклад, команда

```
int *a = (int*) malloc(sizeof(int) * 10);
```

виділяє пам’ять під 10 цілих чисел й адресу початку цієї ділянки пам’яті записує у вказівник `a`.

3) Виділення динамічної пам'яті функцією `calloc`, яка має такий формат:

```
void * calloc(size_t num, size_t size);
```

виділяє блок пам’яті розміром `num`×`size` (під `num` елементів по `size` байтів кожен) і повертає вказівник на виділений блок. Кожен елемент виділеного блока ініціалізується нульовим значенням (на відміну від функції `malloc`). Функція `calloc()` повертає `NULL`, якщо не вистачає пам’яті для виділення нового блока, або якщо значення `num` чи `size` дорівнюють 0.

Виділення пам’яті під 10 дійсних чисел за допомогою функції `calloc()`:

```
double *b = (double*) calloc(10, sizeof(double));
```

Змінити (зменшити чи то збільшити) розмір виділеного раніш блока динамічної пам’яті можна функцією **realloc**, яка має такий формат:

```
void *realloc(*bl, size);
```

при цьому розмір виділеного раніш блока пам’яті з адресою `*bl` змінюється на новий обсяг, який становитиме `size` байтів. Якщо зміна відбулася успішно, функція `realloc()` повертає вказівник на виділену ділянку пам’яті, а інакше повертається `NULL`. Якщо `*bl` є `NULL`, функція `realloc()` виконує такі самі дії, що й `malloc()`. Якщо `size` є 0, виділений за адресою `*bl` блок пам’яті звільнюється – і функція повертає `NULL`. Для ілюстрації роботи функції `realloc()` наведемо приклад змінення розміру раніш виділеної пам’яті під одновимірний масив з 10-ти дійсних елементів до 20-ти елементів:

```
a = (int*) realloc(a, 20);
```

Пам’ять, виділену за допомогою функцій `malloc()` і `calloc()`, слід звільнити за допомогою функції `free()`, наприклад:

```
free(a);
```

Якщо не звільняти динамічно виділену пам’ять, коли вона стає більш не потрібною, у системі може виникнути нестача вільної пам’яті. Іноді це називають "витіком пам’яті".

Використання згаданих функцій разом з вказівниками надає можливість керувати розподілом динамічної пам’яті.

Типи вказівників у Visual C++

У Visual Studio, крім узагальненого і вже розглянутого нами базового поняття нерегульованих вказівників, розглядаються і регульовані вказівники.

Так режим CLR вміє працювати не лише з "рідними" нерегульованими вказівниками, які позначаються *, і пам'ять для яких слід не лише виділяти в так званій некерованій купі, але й звільняти, інакше купа може переповнитись, і процес виконання програмного проекту перерветься. Режим CLR працює і з регульованими вказівниками, керування пам'яттю для яких відбувається під керуванням самого середовища. Для таких вказівників використовують спеціальну позначку: замість символу * застосовують символ ^. Регульовані вказівники повсюдно використовуються при роботі з компонентами на формі. Так у середовищі VC++ існує спеціальна утиліта `gcnew`, яка формує екземпляр відповідного об’єкта, виділяючи йому пам'ять і повертаючи посилання (адресу) на цей екземпляр.

У зв’язку з виникненням більш "просунутих" регульованих вказівників виникла потреба апарата перетворення нерегульованих вказівників до регульованих і навпаки. Цей процес назвали маршалізацією, а для забезпечення цього процесу створили спеціальну бібліотеку.

Доцільність використання вказівників

Правильне розуміння і використання вказівників має велике значення при створенні більшості C++-програм з чотирьох причин:

1) використання вказівників може підвищити ефективність роботи деяких функцій;

2) вказівники надають можливість "бачити" в основній програмі можливі модифікації аргументів у функціях;

3) вказівники використовуються для підтримки системи динамічного виділення пам'яті;

4) вказівники використовуються для підтримки певних структур даних, а саме: зв'язані списки і бінарні дерева.

Крім того, що вказівники – одна з найсильніших сторін С, вони, в той самий час, можуть завдати великої шкоди. Наприклад, неініціалізований чи дикий вказівник може спричинити крах системи, може бути навіть гірше, коли некоректне використання вказівників призводить до невловимих помилок.

Приклади програм

Приклад 1. Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених після першого двозначного числа (якщо двозначних чисел немає, вибрати всі). За допомогою функції поміняти місцями елементи, які стоять поряд: 1 і 2, 3 і 4 тощо.

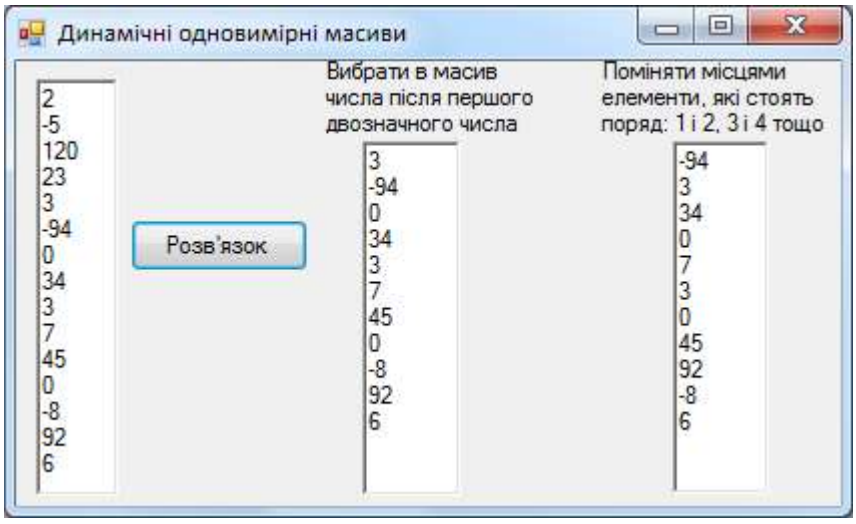
Розв'язок. Якщо організувати створення програмного проекту з формою, це дозволить одночасно бачити введену послідовність чисел (в `richTextBox1`), сформований масив (в `richTextBox2`) і масив з переставленими елементами (в `richTextBox3`).

Програмний код:

```
void f(int a[], int n)
{
  if(n%2) n--; //Якщо кількість елементів непарна, не розглядати останній з них
  for(int i=0; i<n; i+=2)
  { int tmp=a[i];
    a[i]=a[i+1];
    a[i+1]=tmp;
  }
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
  int n=0, i, kol;
  for(i=0; i<richTextBox1->Lines->Length; i++)
  { // Зчитування числа x з richTextBox1. Оскільки двозначними числами можуть
    // бути і від'ємні числа, для спрощення наступної умови беремо модуль числа
    int x=Math::Abs(Convert::ToInt32(richTextBox1->Lines[i]));
    if(x<10 || x>99) continue;
     else { n=i+1; break;} // n – номер першого двозначного числа
  }
  if(!n) MessageBox::Show("Немає двозначних чисел!");
  else
  { // Обчислення kol – кількості елементів масиву
    kol = richTextBox1->Lines->Length - n;
```

```
    // Оголошення динамічного масиву
    int *a = new int[kol];
    // Введення елементів масиву з richTextBox1, починаючи з рядка n
    for(i=0; i<kol; i++)
      a[i]=Convert::ToInt32(richTextBox1->Lines[i+n]);
    // Виведення елементів сформованого динамічного масиву
    richTextBox2->Clear();
    for(i=0; i<kol; i++)
     richTextBox2->AppendText(a[i].ToString()+"\n");
    f(a,kol); // Виклик функції для переставляння елементів масиву
    // Виведення масиву з переставленими елементами
    richTextBox3->Clear();
    for(i=0; i<kol; i++)
      richTextBox3->AppendText(a[i].ToString()+"\n");
    // Звільнення динамічної пам'яті
    delete []a;
  }
}
```

Результати роботи:



Приклад 2. Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, значення яких попадає в проміжок [60, 100]. За допомогою функції визначити мінімальний і максимальний елементи та обчислити середнє арифметичне всіх елементів.

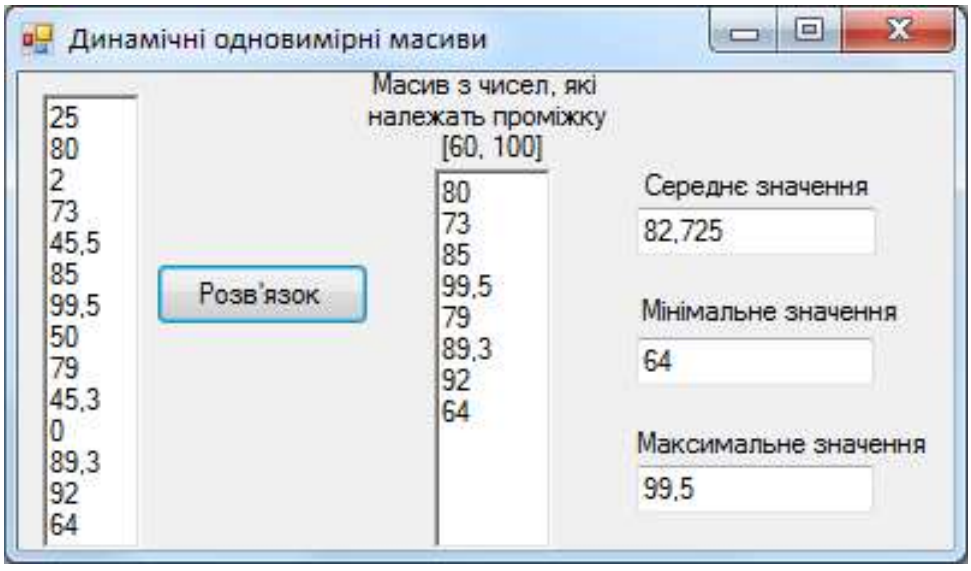
Розв'язок. Щодо практичного застосування уявімо, що початкова послідовність чисел є списком екзаменаційних оцінок, з якого слід відібрати лише ті, які задовольняють умові успішної здачі екзамену. А тому доречним є визначення середнього балу успішності та визначення найкращої і найгіршої екзаменаційної оцінки.

Програмний код:

```
double fun(double a[], int n, double &min, double &max)
{ double s=0;   min=max=a[0];
```

```
  for(int i=0; i<n; i++)
  { s+=a[i];
    if(min>a[i]) min=a[i];
    if(max<a[i]) max=a[i];
  }
  return s/n;
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
  int kol=0, i, j;
  for(i=0; i<richTextBox1->Lines->Length; i++)
  { // Зчитування числа x з richTextBox1
    double x=Convert::ToDouble(richTextBox1->Lines[i]);
    if(x>=60 && x<=100) kol++; // kol – кількість елементів масиву
  }
  if(!kol) MessageBox::Show("Немає чисел з діапазону [60, 100]");
  else
  { double *a = new double[kol];   // Оголошення динамічного масиву
    // Вибірка елементів масиву з richTextBox1
    for(i=j=0; i<richTextBox1->Lines->Length; i++)
    { double x=Convert::ToDouble(richTextBox1->Lines[i]);
      if(x>=60 && x<=100) { a[j]=x; j++; }
    }
    richTextBox2->Clear();
    for(i=0; i<kol; i++)    // Виведення елементів сформованого масиву
      richTextBox2->AppendText(a[i].ToString()+"\n");
    double sr, min, max;
    sr=fun(a,kol,min,max);          // Виклик функції
    textBox1->Text=sr.ToString();  // Виведення результатів
    textBox2->Text=min.ToString();
    textBox3->Text=max.ToString();
    delete []a;                     // Звільнення динамічної пам'яті
} }
```

Результати роботи:

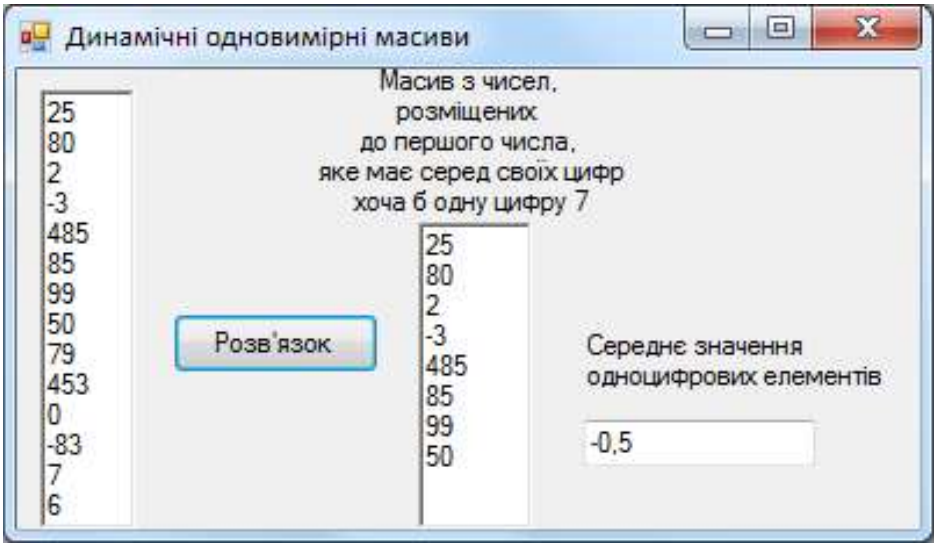


Приклад 3. Ввести деяку послідовність цілих чисел і створити динамічний масив лише з чисел, розміщених до першого числа, яке має серед своїх цифр хоча б одну цифру 7. За допомогою функції обчислити середнє значення одноцифрових елементів масиву.

Програмний код:

```
double fun(int a[], int n)
{
  double s=0; int k=0;
  for(int i=0; i<n; i++)
    if(a[i]>=-9 && a[i]<=9) // перевірка: число одноцифрове?
      { s+=a[i]; k++;  }
  if(k) s/=k;
  return s;
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
  int kol=0, i, n; // kol – кількість елементів масиву
  for(i=0; i<richTextBox1->Lines->Length; i++)
  { // Зчитування числа x з richTextBox1
    int x=Convert::ToInt32(richTextBox1->Lines[i]);
    bool y=false;
    // Пошук сімок у числі. Якщо число має сімку, то kol – номер числа
    while(x)
    { n = x%10; // виділення окремої цифри числа в n
      if(n==7) { y=true; kol=i; break;}
      x /= 10;
    }
    if(y) break;  // Якщо число має сімку, не перевіряти решту чисел
  }
  if(!kol) MessageBox::Show("Немає чисел з сімками");
  else
  {
    int *a = new int[kol]; // Оголошення динамічного масиву
    // Введення елементів масиву з richTextBox1 до першого числа з сімкою
    for(i=0; i<kol; i++)
      a[i]=Convert::ToInt32(richTextBox1->Lines[i]);
    // Виведення елементів сформованого масиву
    richTextBox2->Clear();
    for(i=0; i<kol; i++)
      richTextBox2->AppendText(a[i].ToString()+"\n");
    double sr=fun(a,kol);          // Виклик функції
    textBox1->Text=sr.ToString(); // Виведення результатів
    delete []a;                    // Звільнення динамічної пам’яті
  }
}
```

Результати роботи:



Питання для самоконтролю

1) Що таке вказівник? Наведіть синтаксис оголошення вказівника?
2) Яке тлумачення має значення вказівника `NULL`?
3) В який спосіб можна отримати адресу змінної? Яке призначення операції `&`?
4) Поясніть відмінності між змінними `a` та `b`, оголошення яких має вигляд:
   а) `int a; double b;` б) `int *a; double *b;`
5) Поясніть відмінності динамічного масиву від звичайного.
6) Виберіть правильні оголошення динамічного масиву з 5-ти цілих чисел:
   a) `int a[5];`
   b) `int *a[5];`
   c) `int *a=malloc(5);`
   d) `int *a=(int*) malloc(20);`
   e) `int *a=(int*) malloc(5*sizeof(int));`
   f) `int *a=new int [5];`
   g) `int *a=new [5];`
   h) `int *a=new int (5);`
7) В який спосіб можна звільнити пам'ять, виділену за допомогою оператора `new`? Яка функція застосовується для звільнення пам'яті, виділеної функцією `calloc`?
8) Виберіть правильні команди звільнення пам'яті від динамічного масиву `a` з 5-ти елементів:
   a) `delete a[5];`
   b) `delete a[];`
   c) `delete []a;`
   d) `free (a);`
   e) `free a[5];`
9) У чому полягає відмінність роботи функцій `malloc()` та `calloc()`?
10) Яким є призначення функції `realloc()`?

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи скласти схему алгоритму і написати програму мовою С++ для розв'язання індивідуальних завдань з опрацювання одновимірних масивів з виділенням динамічної пам'яті (завдання вибрати з табл. 15.1).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 15.1

№ вар.	Індивідуальне завдання
1	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з тих чисел, значення яких менше 6-ти. За допомогою функції обчислити середнє арифметичне елементів масиву
2	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з додатних чисел. За допомогою функції визначити мінімальний елемент масиву та його індекс
3	Ввести деяку послідовність дійсних чисел і створити динамічний масив з чисел, розміщених до першого від'ємного числа (якщо від'ємних чисел немає, вибрати всі). За допомогою функції розмістити елементи масиву у порядку зростання значень
4	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з парних ненульових чисел. За допомогою функції визначити мінімальний та максимальний елементи масиву
5	Ввести деяку послідовність чисел і створити динамічний масив з чисел, розміщених до першого числа зі значенням нуль (якщо нулів немає, вибрати всі числа). За допомогою функції обчислити добуток елементів, значення яких за модулем менше 10-ти
6	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, модуль яких не перевищує 8. За допомогою функції обчислити середнє арифметичне мінімального і максимального елементів масиву
7	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених до першого тризначного числа (якщо тризначних чисел немає, вибрати всі). За допомогою функції обчислити середнє арифметичне непарних елементів
8	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з двозначних чисел. За допомогою функції обчислити кількість та суму парних елементів масиву
9	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених до першого двозначного числа. За допомогою функції обчислити добуток непарних елементів масиву

Закінчення табл. 15.1

№ вар.	Індивідуальне завдання
10	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених після першого тризначного числа (якщо тризначних чисел немає, видати про це повідомлення). За допомогою функції обчислити середнє арифметичне елементів масиву, кратних 3
11	Ввести деяку послідовність дійсних чисел і створити динамічний масив з чисел, розміщених після першого числа зі значенням 0 (якщо нулів немає, видати про це повідомлення). За допомогою функції обчислити суму елементів, абсолютне значення яких не перевищує 10
12	Ввести деяку послідовність дійсних чисел і створити динамічний масив з чисел, розміщених після першого від'ємного числа (якщо від'ємних чисел немає, видати про це повідомлення). За допомогою функції поміняти місцями мінімальний і максимальний елементи масиву
13	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, значення яких належить проміжку [10, 25]. За допомогою функції визначити кількість елементів, значення яких більше за значення першого елемента масиву
14	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з непарних чисел. За допомогою функції замінити всі від'ємні елементи нулями
15	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, які перевищують свій порядковий номер. За допомогою функції розмістити елементи масиву у зворотному порядку
16	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, модуль яких не попадає в проміжок (20, 40]. За допомогою функції обчислити кількість елементів, що перевищують середнє арифметичне всіх елементів
17	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з ненульових чисел. За допомогою функції визначити найбільший з парних елементів
18	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, модуль яких попадає в проміжок [5, 50). За допомогою функції визначити мінімальний із додатних елементів
19	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених до першого від'ємного двозначного числа (якщо від'ємних двозначних чисел немає вибрати всі). За допомогою функції замінити мінімальний і максимальний елементи значенням середнього арифметичного всіх елементів
20	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з чисел, кратних 3. За допомогою функції обчислити середнє арифметичне елементів, значення яких більше значення останнього елемента масиву
21	Ввести деяку послідовність цілих чисел і створити динамічний масив лише з чисел, значення яких не перевищують 100. За допомогою функції обчислити суму тільки двоцифрових елементів

Закінчення табл. 15.1

№ вар.	Індивідуальне завдання
22	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених після першого двозначного числа. За допомогою функції обчислити середнє арифметичне елементів, кратних 5
23	Ввести деяку послідовність цілих додатних чисел і створити динамічний масив з чисел, які мають серед своїх цифр хоча б одну цифру 2. За допомогою функції обчислити добуток одноцифрових елементів масиву
24	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, які завершуються цифрою 5. За допомогою функції обчислити середнє арифметичне двозначних елементів
25	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, розміщених після першого парного числа (якщо парних чисел немає, видати про це повідомлення). За допомогою функції поміняти місцями мінімальний елемент з першим
26	Ввести деяку послідовність цілих додатних чисел і створити динамічний масив з чисел, які мають серед своїх цифр хоча б одну цифру 1. За допомогою функції поміняти місцями першу половину масиву з другою
27	Ввести деяку послідовність дійсних чисел і створити динамічний масив з чисел, розміщених до першого числа, модуль якого більше 100. За допомогою функції визначити максимальний елемент та його індекс
28	Ввести деяку послідовність дійсних чисел і створити динамічний масив з чисел, які відрізняються від числа 5 не більше ніж на 10. За допомогою функції визначити максимальний з від’ємних елементів масиву та його номер
29	Ввести деяку послідовність цілих чисел і створити динамічний масив з чисел, які завершуються цифрою 1. За допомогою функції замінити елементи, кратні 3, на 0
30	Ввести деяку послідовність дійсних чисел і створити динамічний масив лише з чисел, значення яких належить проміжку (–20, 10]. За допомогою функції обчислити середнє арифметичне від’ємних елементів

Лабораторна робота № 16

Вказівники і динамічна пам'ять при опрацюванні двовимірних масивів

Мета роботи: набути практичних навиків програмного використання вказівників та динамічної пам’яті при опрацюванні двовимірних масивів.

Теоретичні відомості

Вказівники на багатовимірні масиви

Вказівники на багатовимірні масиви у мові C++ – це масиви масивів, тобто такі масиви, елементами яких є масиви. При оголошені таких масивів у пам’яті комп’ютера створюється декілька різних об’єктів. Приміром, при виконанні оголошення двовимірного масиву `int arr[4][3]` у пам’яті виділяється ділянка для зберігання значення змінної `arr`, яка є вказівником на масив з чотирьох вказівників. Для цього масиву з чотирьох вказівників теж виділяється пам’ять. Кожний з цих чотирьох вказівників містить адресу масиву з трьох елементів типу `int`, і, отже, у пам’яті комп’ютера виділяється чотири ділянки для зберігання чотирьох масивів чисел типу `int`, кожна з яких складається з трьох елементів. Такий розподіл пам’яті показано на схемі:

```
   arr
    ↓
arr[0]    →    arr[0][0]   arr[0][1]   arr[0][2]
arr[1]    →    arr[1][0]   arr[1][1]   arr[1][2]
arr[2]    →    arr[2][0]   arr[2][1]   arr[2][2]
arr[3]    →    arr[3][0]   arr[3][1]   arr[3][2]
```

Отже, оголошення `arr[4][3]` породжує в програмі три різних об’єкти: вказівник з ідентифікатором `arr`, безіменний масив з чотирьох вказівників і безіменний масив з дванадцяти чисел типу `int`. Для доступу до безіменних масивів використовуються адресні вирази з вказівником `arr`. Доступ до елементів масиву вказівників здійснюється із зазначенням одного індексного виразу у формі `arr[2]` чи `*(arr+2)`. Для доступу до елементів двовимірного масиву чисел типу `int` має бути використано два індексні вирази у формі `arr[1][2]` або еквівалентних їй `*(*(arr+1)+2)` та `(*(arr+1))[2]`.

Нагадаємо, що елементи двовимірних масивів розміщуються у пам’яті підряд і між його рядками немає ніяких проміжків. Такий порядок дає можливість звертатися до будь-якого елемента багатовимірного масиву, використовуючи адресу його початкового елемента та лише один індексний вираз. Так для матриці `arr` звертання `*(arr)` є посиланням на елемент `arr[0][0]`, звертання `*(arr+2)` – на елемент `arr[0][2]`, а звертання `*(arr+3)` – на елемент `arr[1][0]` і т. д. Тобто, застосовуючи оператори циклу, можна організувати поелементне опрацювання елементів матриці, наприклад введення усієї матриці з компонента `dataGridView1`:

```
for(int i=0; i<4; i++)
for(int j=0; j<3; j++)
  *(arr+i*4+j)=Convert::ToInt32(dataGridView1[j,i]->Value);
```

Тут вираз `*(arr+i*4+j)` є аналогічний до `arr[i][j]`.

Динамічна пам'ять для матриць

Основний спосіб роботи з динамічними матрицями базується на використанні подвійних вказівників. Для того, щоб працювати з динамічними матрицями, доцільно представити матрицю як масив векторів (масив, що містить вказівники на рядки матриці). Перед початком роботи з такою матрицею треба спочатку виділити пам'ять під масив вказівників на рядки, а лише тоді виділяти пам'ять для зберігання самих даних. Наприклад, оголошення динамічної матриці 7×8 дійсних чисел може бути таким:

```
double **a = new double*[7];          // сім рядків матриці
for (int i=0; i<7; i++)
  a[i] = new double[8];               // по вісім стовпців
```

Після використання матриці виділена для неї пам'ять звільняється у зворотній послідовності:

```
for(int i=0; i<7; i++) delete [] a[i];
delete []a;
```

Тобто при виділенні пам’яті для цієї матриці спочатку оголошується вказівник другого порядка `double **a`, який посилається на масив вказівників `double*[7]` (де 7 – розмір масиву). Після цього в циклі для кожного рядка матриці виділяється по вісім елементів.

Звертання до елементів динамічної матриці здійснюється так само, як і до елементів статичної матриці – `a[i][j]`.

Наведемо приклад програми з динамічним виділенням пам’яті для матриці, розмірність якої (`m×n`) вводить користувач. Значення елементів цієї матриці заповнюються в програмі випадковими числами з діапазону від 1 до 100 і виводяться на екран.

Для генерації випадкових чисел у мові С існує спеціальна функція `rand()`, яка повертає псевдовипадкове ціле число у діапазоні від 0 до вказаного значення. Приміром, команда для надання цілій змінній псевдовипадкового значення від 0 до 99 буде такою:

```
int x=rand() % 100;
```

Тобто вираз `rand()%n` сформує псевдовипадкове ціле число з проміжку від 0 до `n-1`. Щоб отримати псевдовипадкове число не від 0 до `n`, а з іншого проміжку (приміром `[a; a+n-1]`), треба скористатись виразом `rand()%n+a`. Причому, `a` може бути як додатним, так і від’ємним. Наприклад, вираз `(rand() % 31)+20` генеруватиме числа від 20 до 50, а вираз `(rand() % 101)-50` – числа від –50 до 50.

Перед використанням цієї функції треба ініціалізувати генератор випадкових чисел функцією `srand()` для того, щоб кожного разу формувалися різні, а не одні й ті самі числа. Аргументом функції `srand()` можна задати функцію `time()`, викликану з нульовим вказівником типу `time_t`, яка повертає кількість

секунд від 1 січня 1970 року і тим самим задає випадкове зерно для генерації чисел. Функції rand() та srand() прописані в заголовному файлі <cstdlib>, а time() – у файлі <ctime>. Наприклад, команди

```
time_t t;
srand((unsigned) time(&t));
for(int i=0; i<10; i++) printf("%d\n", rand() % 100);
```

генеруватимуть послідовність з 10-ти випадкових цілих чисел від 0 до 99. Подібна конструкція для дійсних чисел може бути такою:

```
srand(time(NULL));        // можна і так:  srand(time(0));
for(int i=0; i<10; i++)
  cout<<setw(4)<<setprecision(2)<<(rand()%100)/double((rand()%10))<<endl;
```

Приклад програми з динамічним виділенням пам'яті для матриці випадкових чисел, розмірність якої (m×n) вводить користувач:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{ setlocale(0,".1251");
  srand(time(NULL));              // генерація випадкових чисел
  int m, n;                       // змінні для розмірності матриці
  cout<<"Введіть розмірність матриці через пробіл  ";  cin>>m>>n;
  // динамічне створення матриці дійсних чисел розміром m×n
  double **a = new double*[m];  // n рядків матриці
  for (int i=0; i<m; i++) a[i] = new double[n]; // по m стовпців
  // заповнення матриці випадковими дійсними числами від 1 до 10
  for (int i=0; i<m; i++)
   for (int j=0; j<n; j++)
    a[i][j] = (rand() % 100 + 1) / double((rand() % 10 + 1));
  // виведення матриці
  for (int i=0; i<m; i++)
  { for (int j=0; j<n; j++)
      cout << setw(4) << setprecision(2) << a[i][j] << "   ";
    cout << endl;
  }
  for (int i=0; i<m; i++) delete []a[i];  // звільнення динамічної пам'яті
  delete []a;
  system("pause>>void");
  return 0;
}
```

```
Уведіть розмірність матриці через пробіл  3 7
 9.5    8.8   0.89    1.3     35    3.2    8.6
 2.3     11    5.5     21    0.8    9.1     19
  58     13     13    8.8     14     20    9.8
```

При виведенні цієї матриці було використано модифікатор setw, який дозволив задавати ширину (кількість символьних позицій) кожного виведеного

числа. Так, модифікатор `setw(4)` для кожного елемента відвів максимум по чотири позиції, що дозволило вирівняти по стовпцях елементи різного розміру. Якщо елемент мав меншу, аніж чотири знаки ширину, перед ним було виведено пробіли. Використаний при виведенні маніпулятор `setprecision(2)` дозволив обмежити кількість знаків після десяткової крапки максимум двома знаками. Саме для цих засобів було долучено файл `iomanip`, а докладно вони були розглянуті в лабораторній роботі № 1.

Приклади програм

Приклад 1. Ввести цілочисельну матрицю 6×4 і створити нову матрицю з тих рядків введеної матриці, які не містять нульових елементів.

Розв'язок. На відміну від введеної матриці, створювана матриця буде динамічною, оскільки кількість її рядків заздалегідь є невідома. Для обчислення кількості нульових елементів кожного рядка організуємо тимчасовий допоміжний масив `r`. Кількість рядків створюваної матриці визначатимемо як кількість ненульових елементів масиву `r` і у нову матрицю будемо копіювати лише ті рядки пведеної, для яких `r=0`.

Програмний код:

```
#include <iostream>
using namespace std;
int main()
{
  setlocale(0,".1251");
  int a[6][4], i, j, rowc=0, k=0;
  cout<<"Введіть матрицю з 6-ти рядків і 4-х стовпців:"<<endl;
  int r[6]={0};  // r – масив кількості нульових елементів рядків матриці
  for(i=0; i<6; i++)
   for(j=0; j<4; j++)
   { cin>>a[i][j];
     if(a[i][j]==0) r[i]++;
   }
  for(i=0; i<6; i++)  // Обчислення кількості рядків без нульових елементів
    if(r[i]==0) rowc++;  // r[i]=0  означає, що в і-му рядку немає нулів
  int **b=new int *[rowc];  // Виділення пам'яті для динамічної матриці b
  for(i=0; i<rowc; i++) b[i]=new int [4];
  for(i=0; i<6; i++)
   if(r[i]==0)            // Якщо в і-му рядку нулів немає, відбувається
    { for(j=0;j<4;j++) b[k][j]=a[i][j]; // копіювання рядка у нову матрицю
      k++;                // і збільшення індексу рядка нової матриці.
    }
  cout<<"\nМатриця, в рядках якої немає нульових елементів:"<<endl;
  for(i=0; i<rowc; i++)
  { for(j=0; j<4; j++) cout<<b[i][j]<<"\t";
    cout << endl;
  }
```

```
  // Звільнення пам'яті від динамічної матриці
  for(int i=0; i<rowc; i++)
    delete [] b[i];
  delete []b;
  system ("pause>>void");
  return 0;
}
```

Результати роботи:

```
Введіть матрицю з 6-ти рядків і 4-х стовпців:
23      7       90      0
-6      0       88      45
33      78      9       4
12      4       8       90
0       3       0       -7
50      7       9       7

Матриця, в рядках якої немає нульових елементів:
33      78      9       4
12      4       8       90
50      7       9       7
```

Приклад 2. Ввести матрицю 3×7 дійсних чисел та створити нову матрицю з тих стовпців введеної матриці, які містять нульові елементи.

Програмний код:

```
#include <iostream>
using namespace std;
int main()
{ setlocale(0,".1251");
  double a[3][7]; int i, j;
  cout<<"Введіть матрицю з 3-х рядків і 7-ми стовпців:"<<endl;
  for(i=0; i<3; i++)
   for(j=0; j<7; j++) cin>>a[i][j];
  int col=0;      // col – кількість стовпців з нульовими елементами
  int r[7]={0};  // r – масив кількості нульових елементів стовпців матриці
  for(j=0; j<7; j++)
  { for(i=0; i<3; i++)        // Обчислення
      if(a[i][j]==0) r[j]++; // кількості нулів у стовпці та
    if(r[j]!=0) col++;        // кількості стовпців з нульовими елементами
  }
  double **b=new double*[3];  // Виділення пам'яті
  for(i=0; i<3; i++)           // для динамічної матриці
    b[i]=new double [col];
  int k=0;       // k – індекс стовпця нової матриці (з нульовими елементами)
  for(j=0; j<7; j++)
   if(r[j]!=0)       // Якщо в j-му стовпці є нулі, скопіювати
    { for(i=0;i<3;i++) b[i][k]=a[i][j]; // стовпець у нову матрицю
      k++;           // і збільшити індекс стовпця нової матриці
    }
```

```
  if(!col) cout<<"\nНемає стовпців з нульовими елементами";
  else
  cout<<endl<<col<<" стовпці(в) з нульовими елементами";
  cout<<"\nМатриця, у стопвцях якої є нульові елементи:"<<endl;
  for(i=0; i<3; i++)
  { for(j=0; j<col; j++) cout<<b[i][j]<<"\t";
    cout << endl;
  }
  for(int i=0; i<3; i++)
    delete [] b[i];      // Звільнення пам'яті від динамічної матриці
  delete []b;
  system ("pause>>void");
  return 0;
}
```

Результати роботи:

```
Введіть матрицю з 3-х рядків і 7-ми стовпців:
45      7.8     0       -7      2.8     30      -5
0       5.8     5.7     0       12      -7      0
0.1     -5      0       23      6       4.8     41

4 стовпці(в) з нульовими елементами
Матриця, у стопвцях якої є нульові елементи:
45      0       -7      -5
0       5.7     0       0
0.1     0       23      41
```

Питання для самоконтролю

1) Виберіть правильне оголошення динамічної матриці 5×4 цілих чисел:

a) `int a[5][4];`

b) `int *a[5][4];`

c)
```
int *a=new int [5][4];
  for(int i=0; i<5; i++) a[i]=new int [4];
```

d) `int **a=new int*[5];`

e)
```
int **a=new int *[4];
  for(int i=0; i<4; i++) a[i]=new int [5];
```

2) Виберіть правильний варіант звільнення пам'яті від динамічної матриці `a` з 5×4 цілих елементів:

a) `delete a[5][5];`

b) `delete a[][];`

c) `delete [][]a;`

d)
```
for(int i=0; i<5; i++) delete [] a[i];
  delete []a;
```

e)
```
for(int i=0; i<4; i++) delete [] a[i];
  delete []a;
```

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи скласти схему алгоритму і написати програму мовою С++ для розв'язання індивідуальних завдань з опрацювання двовимірних масивів з виділенням динамічної пам'яті (завдання вибрати з табл. 16.1).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 16.1

№ вар.	Індивідуальне завдання
1	Ввести матрицю 10×3 дійсних чисел і створити нову матрицю з тих рядків введеної матриці, які не містять від'ємних елементів
2	Ввести цілочисельну матрицю 3×9 і створити нову матрицю з тих стовпців введеної матриці, які містять парні елементи
3	Ввести матрицю 9×3 дійсних чисел і створити нову матрицю з тих рядків введеної матриці, які містять елементи лише в діапазоні [50, 100]
4	Ввести цілочисельну матрицю 3×8 і створити нову матрицю з тих стовпців введеної матриці, які містять елементи, кратні 3
5	Ввести цілочисельну матрицю 10×3 і створити нову матрицю з тих рядків введеної матриці, які містять хоча б один елемент – двійку
6	Ввести матрицю 3×9 дійсних чисел і створити нову матрицю з тих стовпців введеної матриці, які не мають елементи менші 10-ти
7	Ввести цілочисельну матрицю 8×3 і створити нову матрицю з тих рядків введеної матриці, які містять хоча б один елемент, кратний 100
8	Ввести цілочисельну матрицю 5×9 і створити нову матрицю з тих стовпців введеної матриці, які не містять жодного елемента 10
9	Ввести цілочисельну матрицю 9×5 і створити нову матрицю з тих рядків введеної матриці, які не містять парні елементи
10	Ввести матрицю 3×6 дійсних чисел і створити нову матрицю з тих стовпців введеної матриці, які містять від'ємні елементи
11	Ввести цілочисельну матрицю 10×4 і створити нову матрицю з тих рядків введеної матриці, які містять елементи, кратні 10
12	Ввести матрицю 4×9 дійсних чисел і створити нову матрицю з тих стовпців введеної матриці, які мають елементи більші за 100
13	Ввести цілочисельну матрицю 8×4 і створити нову матрицю з тих рядків введеної матриці, які не містять жодного елемента 5 чи 25
14	Ввести матрицю 3×10 дійсних чисел і створити нову матрицю з тих стовпців введеної матриці, які мають елементи за модулем менші 7-ми
15	Ввести цілочисельну матрицю 7×3 і створити нову матрицю з тих рядків введеної матриці, які містять хоча б один елемент – 7 або 13
16	Ввести цілочисельну матрицю 4×10 і створити нову матрицю з тих стовпців введеної матриці, які містять хоча б один елемент – одиницю
17	Ввести матрицю 8×3 дійсних чисел і створити нову матрицю з тих рядків

№ вар.	Індивідуальне завдання
	введеної матриці, які не містять елементів більших за 100

Закінчення табл. 16.2

№ вар.	Індивідуальне завдання
18	Ввести цілочисельну матрицю 4×9 і створити нову матрицю з тих стовпців введеної матриці, які не містять елементи, кратні 10
19	Ввести матрицю 9×5 дійсних чисел і створити нову матрицю з тих рядків введеної матриці, які мають елементи менші 5-ти
20	Ввести цілочисельну матрицю 4×9 і створити нову матрицю з тих стовпців введеної матриці, які містять хоча б один елемент 2 або –1
21	Ввести цілочисельну матрицю 7×3 і створити нову матрицю з тих рядків введеної матриці, які не містять жодного елемента 7
22	Ввести цілочисельну матрицю 4×8 і створити нову матрицю з тих стовпців введеної матриці, які містять непарні елементи
23	Ввести матрицю 10×3 дійсних чисел і створити нову матрицю з тих рядків введеної матриці, які не мають елементи за модулем менші за 9
24	Ввести матрицю 3×8 дійсних чисел і створити нову матрицю з тих стовпців введеної матриці, які не містять елементи з діапазону (10, 20]
25	Ввести цілочисельну матрицю 8×3 і створити нову матрицю з тих рядків введеної матриці, які не містять елементи, кратні 3
26	Ввести матрицю 5×9 дійсних чисел і створити нову матрицю з тих стовпців введеної матриці, які містять елементи, дійсна частина яких дорівнює 0
27	Ввести цілочисельну матрицю 7×4 і створити нову матрицю з тих рядків введеної матриці, які не містять непарні елементи
28	Ввести цілочисельну матрицю 3×9 і створити нову матрицю з тих стовпців введеної матриці, які містять хоча б один елемент, кратний 20
29	Ввести матрицю 7×5 дійсних чисел і створити нову матрицю з тих рядків введеної матриці, які не містять елементи, дійсна частина яких дорівнює 0
30	Ввести цілочисельну матрицю 5×9 і створити нову матрицю з тих стовпців введеної матриці, які не містять жодного елемента 11 або 22

Лабораторна робота № 17

Робота з символьними даними

Мета роботи: набути практичних навиків програмного опрацювання символьних даних.

Теоретичні відомості

Символьний тип даних у C++

Символами вважаються: великі й малі латинські літери, великі й малі літери кирилиці, цифри, пробіл, розділові знаки ('.', ',', ';', ':', '!', '?', '–'), знаки арифметичних дій ('+', '–', '*', '/', '='), службові символи, що відповідають клавішам *Enter*, *Esc*, *Tab* тощо. У C++ значення символьних констант записуються в одинарних лапках: '3', 'f', '+', '%'.

Існує єдиний міжнародний стандарт – так звана таблиця ASCII-кодів (American Standard Code for Information Interchange – американський стандартний код для обміну інформацією). Символи ASCII мають коди від 0 до 127, тобто значення першої половини можливих значень байта, хоча часто кодами ASCII називають всю таблицю з 256 символів. Перші 128 ASCII-кодів є єдині для всіх країн, а коди від 128 до 255 називають розширеною частиною таблиці ASCII, де залежно від країни розміщується національний алфавіт і символи псевдографіки. У різних версіях C++ коди символів національного алфавіту можуть мати різні значення.

Символьний тип у C++ зветься **char**. Наприклад, при оголошенні змінних

```
char c, s, g;
```

в оперативній пам'яті для кожної з цих трьох змінних буде відведено по одному байту. Коли символьні змінні набудуть певних значень, то комп'ютер запише в пам'ять не самі символи, а їхні коди (у вигляді цілих чисел). Наприклад, замість літери 'A' зберігатиметься її код 65. Тому, якщо присвоїти символьній змінній певне ціле число, то C++ сприйме його як код символу з таблиці ASCII-кодів.

Символи можна порівнювати. Більшим вважається той символ, у якого код є більший, тобто символ, розміщений у таблиці ASCII-кодів пізніше. Наприклад: 'a' < 'h', 'A' < 'a'.

Специфіка мови C++ при опрацюванні символьних даних полягає в тому, що за замовчуванням тип є знаковим, тобто зберігає значення кодів символів від –128 до 127. При чому від'ємні значення мають коди розширеної частини таблиці ASCII. Наступний фрагмент програмного коду дозволить побачити як C++ інтерпретує символи кирилиці:

```
setlocale(LC_ALL,".1251");
for (int i='А'; i<='я'; i++)
{  if(i%8==0) puts("\n");
   printf("%c = %3i  ",(char)i, i);
}
```

Функції для роботи з символами

У C++ існують спеціальні функції для роботи з символьними даними.

Функція	Призначення
`tolower()`	повертає символ у нижньому регістрі
`toupper()`	повертає символ у верхньому регістрі
Наступні функції перевіряють належність символу до множини:	
`isalnum()`	латинських літер і цифр ('A' – 'Z', 'a' – 'z', '0' – '9')
`isalpha()`	латинських літер ('A' – 'Z', 'a' – 'z')
`iscntrl()`	керувальних символів (з кодами 0..31 й 127)
`isdigit()`	цифр ('0' – '9')
`isgraph()`	видимих символів, тобто не є відповідними клавішам *Esc*, *Tab* тощо
`islower()`	латинських літер нижнього регістру ('a' – 'z')
`isprint()`	друкованих символів (`isgraph()` + пробіл)
`isupper()`	літер верхнього регістру ('A' – 'Z')
`ispunct()`	знаків пунктуації
`isspace()`	символів-розділювачів

Примітка. Вищезазначені функції перевірки й перетворення регістру працюють лише з латинськими літерами. Спроба використати ці функції для символів кирилиці спричинить виведення повідомлення про помилку. Тому для роботи з літерами кирилиці слід використовувати перевірку належності символу до відповідного символьного проміжку, а для перетворення регістру – зменшення чи то збільшення коду символу на різницю кодів між великими та малими літерами (для більшості літер вона становить 32).

В C++ рядки розглядають як масиви, елементи якого мають тип `char`. Рядок може містити символи літер, цифр і спеціальних символів, які записують у подвійних лапках. Ім’я рядка є константним вказівником на перший символ. Нумерація символів у рядку починається з 0. Останній символ рядка – службовий символ, так званий, завершальний нуль-символ '\0', який є ознакою кінця рядка. При оголошенні рядка слід вказати максимальну кількість символів, які будуть зберігатися у рядку, при чому слід враховувати наявність наприкінці рядка нуль-термінатора і відводити додатковий байт для нього:

```
char s[10]; // Рядок s може містити до 9-ти символів
```

Рядок при оголошенні можна ініціалізувати початковим значенням, наприклад так:

```
char s[10] = "abcdef";
```

При цьому перші шість символів рядка набудуть значень кодів зазначених символів, а решта чотири символи – значення '\0'. До речі, якщо рядок при оголошенні ініціалізується, то зовсім не обов'язково вказувати його розмір в квадратних дужках:

```
char s[] = "abcdef";
```

При цьому розмір рядка визначається автоматично, а в кінець рядка дописується символ '\0'.

Введення-виведення символьних даних

Для введення одного символу в консолі існує функція `getch()` з бібліотеки `conio.h`.

```
char c = getch();
getch();
```

Оскільки ця функція призупиняє виконання програми й очікує на введення символу, тобто на натиснення будь-якої клавіші, її доволі часто використовують наприкінці консольних програм перед `return 0`, щоб призупинити програму і надати користувачу можливість прочитати результати.

Для введення й виведення рядків у консолі, крім вже знайомих Вам потокових команд C++ `cin>>` та `cout<<` (з бібліотеки `iostream.h`), використовують С-функції `gets/puts` та `scanf/printf` (з бібліотеки `stdio.h`).

1) Специфіка потокової команди введення `cin>>` полягає у тому, що вона вводить послідовність символів до першого пробілу або іншого розділювача, тобто цю команду недоречно використовувати для введення речень з пробілами.

2) Функція `puts(s)` виводить рядок `s` на екран, замінюючи нуль-символ на *Enter*.

3) Функція `gets(s)` зчитує символи з клавіатури до появи символу переведення рядка *Enter* і записує їх у рядок `s` (власне символ *Enter* до рядка не долучається, а замість нього записується нуль-символ), наприклад:

```
const int n=80;
char s[n];
gets(s);
puts(s);
```

Замість функції `gets()` вважається більш безпечним застосовувати функцію `gets_s()`, оскільки вона дозволяє контролювати кількість зчитуваних символів:

```
gets_s(s);
gets_s(s,80);
```

4) Функції форматного введення-виведення **`scanf`** та **`printf`** мають схожий формат і дозволяють вводити не лише рядки, а й числові дані, залежно від вказаного формату:

`scanf` (<*формат*>, < *список_змінних*>);

`printf` (<*формат*>, <*список_змінних*>);

де: *формат* − рядок специфікаторів формату у подвійних лапках. Найбільш

поширеними специфікаторами є: `%s` – для рядка символів, `%i` – для цілих чисел, `%f` – для дійсних чисел у фіксованому форматі;

список_змінних – послідовність розділених комами змінних, значення яких вводиться або виводяться.

Так, функція `printf("Результат: ", str)` виводить рядок `str`, а функція `scanf("%s %i", str, &kol)` вводить значення рядка `str` і цілої змінної `kol`.

Для роботи з рядками існують спеціальні функції, серед яких:

`strlen(s)` – обчислення довжини рядка `s` (нуль-символ не враховується);

`strcat(s1, s2)` – приєднання рядка `s2` до кінця рядка `s1`.

Інші функції для роботи з рядками будуть розглянуті у наступній лабораторній роботі.

Встановлення мовних стандартів

Ще однією особливістю Visual C++ є те, що в консольному режимі порізному інтерпретуються коди розширеної частини таблиці ASCII для введених символьних даних і набраного у програмі тексту. І саме тому текст кирилицею (українською або російською мовою) в консолі виводиться у вигляді абракадабри. Для визначення мовного стандарту (країна або регіон і мова) при роботі з розширеними символами існує спеціальна функція `setlocale()` з бібліотеки `locale.h`, формат якої має вигляд:

```
char *setlocale(int type, const char *locale);
```

де `type` – аргумент, який вказує на те, які саме дані мовного стандарту слід змінити: `LC_ALL` – усі, `LC_MONETARY` – формат грошових значень, `LC_NUMERIC` – формат числових значень з десятковою точкою або комою, `LC_TIME` – формат дати і часу тощо;

`locale` – аргумент, який задає мовний стандарт у вигляді рядка із зазначенням мови, коду країни і/або кодової сторінки у форматі: `"<мова>_<країна>.<кодова сторінка>"`, при чому всі три складові зазначати не обов'язково. Наприклад, команди `setlocale(LC_ALL,".1251")`, `setlocale(0,"Ukrainian.1251")`, `setlocale(0, "Ukrainian_Ukraine.1251")` і `setlocale(0,"uk-UA.1251")` для України є тотожними, а для США тотожними є використання команд `setlocale(0,"English")`, `setlocale(LC_ALL,"English_United States.1252")` і `setlocale(LC_ALL, "en-US")`, при чому перша з форм є найбільш рекомендованою.

Набір доступних імен мовних стандартів, назв мов, кодів країн і кодових сторінок можна побачити за Інтернет-посиланням: http://msdn.microsoft.com/ru-RU/goglobal/bb896001.aspx. Приміром, функція `setlocale(0,"French_Canada.1252")` встановлює мовний стандарт "французький (Канада)" з кодовою сторінкою 1252.

Якщо в якості значення кодової сторінки аргументу `locale` вказати `".OCP"`, то встановлюватиметься кодова сторінка за замовчуванням.

Оскільки мова C++ опрацьовує символ як один байт, то спроба встановлення кодової сторінки UTF-7 чи UTF-8, які потребують два байти для кожного символу, завершиться помилкою.

Функція `setlocale()` повертає вказівник на рядок, асоційований з параметром `type`. При виникненні помилки функція поверне нульовий вказівник.

Розглянемо на прикладі доцільність почергового використання різних кодувань (команди `setlocale(0,".1251")` і `setlocale(0,".OCP")`) у програмному коді для коректного виведення літер кирилиці, оскільки без використання цих команд кирилиця виводитиметься у вигляді абракадабри. Далі наведено три варіанти програмного коду, а праворуч від них – вигляд виведеного тексту:

1)
```
char s[100];
puts("Введіть рядок s =");
gets_s(s);
puts("Введений Вами рядок s =");
puts(s);
```

```
тхфіЄЧ Ё фюъ s =
Мова програмування C++
тхфхэшщ тръш Ё фюъ s =
Мова програмування C++
```

2)
```
char s[100];
setlocale(LC_ALL,".1251");
puts("Введіть рядок s =");
gets_s(s);
puts("Введений Вами рядок s =");
puts(s);
```

```
Введіть рядок s =
Мова програмування C++
Введений Вами рядок s =
?Rў  YaR?a ґrў --п C++
```

3)
```
char s[100];
setlocale(LC_ALL,".1251");
puts("Введіть рядок s =");
setlocale(LC_ALL,"Ukrainian_Ukraine.OCP");
gets_s(s);
setlocale(0,".1251");
puts("Введений Вами рядок s =");
setlocale(0,".OCP");
puts(s);
```

```
Введіть рядок s =
Мова програмування C++
Введений Вами рядок s =
Мова програмування C++
```

Отже, для виведення на екран текстових повідомлень з кирилицею слід використовувати кодування Windows 1251 (команда `setlocale(LC_ALL,".1251")` або `setlocale(0,".1251")`, а для виведення раніш введених символьних даних слід повернутися до початкових налаштувань командою `setlocale(0,".OCP")`.

Приклади програм

Приклад 1. Вивести всі символи та коди таблиці ASCII.

Текст програми:

```
#include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;

int main()
{
   setlocale(LC_ALL,".1251");
```

```
    for (int i=32; i<256; i++)
    { if (i%4==0) printf("\n");
      printf("%3i = %c \t",i,(char)i);
    }
    getch();
    return 0;
}
```

Результати роботи:

```
 32 =                33 = !              34 = "              35 = #
 36 = $              37 = %              38 = &              39 = '
 40 = (              41 = )              42 = *              43 = +
 44 = ,              45 = -              46 = .              47 = /
 48 = 0              49 = 1              50 = 2              51 = 3
 52 = 4              53 = 5              54 = 6              55 = 7
 56 = 8              57 = 9              58 = :              59 = ;
 60 = <              61 = =              62 = >              63 = ?
 64 = @              65 = A              66 = B              67 = C
 68 = D              69 = E              70 = F              71 = G
 72 = H              73 = I              74 = J              75 = K
 76 = L              77 = M              78 = N              79 = O
 80 = P              81 = Q              82 = R              83 = S
 84 = T              85 = U              86 = V              87 = W
 88 = X              89 = Y              90 = Z              91 = [
 92 = \              93 = ]              94 = ^              95 = _
 96 = `              97 = a              98 = b              99 = c
100 = d             101 = e             102 = f             103 = g
104 = h             105 = i             106 = j             107 = k
108 = l             109 = m             110 = n             111 = o
112 = p             113 = q             114 = r             115 = s
116 = t             117 = u             118 = v             119 = w
120 = x             121 = y             122 = z             123 = {
124 = |             125 = }             126 = ~             127 = ⌂
128 = ?             129 = ?             130 = ,             131 = ?
132 = "             133 = :             134 = ┼             135 = ╪
136 = ?             137 = ‰             138 = ?             139 = ?
140 = ?             141 = ?             142 = ?             143 = ?
144 = ?             145 = '             146 = '             147 = "
148 = "             149 =   150 = -               151 = -
152 = ?             153 = T             154 = ?             155 = >
156 = ?             157 = ?             158 = ?             159 = ?
160 =               161 = ў             162 = Ў             163 = ?
164 = ¤             165 = ?             166 = ¦             167 = §
168 = Ё             169 = c             170 = Є             171 = «
172 = ¬             173 = -             174 = R             175 = Ї
176 = °             177 = +             178 = ?             179 = ?
180 = ?             181 = ч             182 = ¶             183 = ·
184 = ё             185 = №             186 = е             187 = »
188 = ?             189 = ?             190 = ?             191 = ї
192 = А             193 = Б             194 = В             195 = Г
196 = Д             197 = Е             198 = Ж             199 = З
200 = И             201 = Й             202 = К             203 = Л
204 = М             205 = Н             206 = О             207 = П
208 = Р             209 = С             210 = Т             211 = У
212 = Ф             213 = Х             214 = Ц             215 = Ч
216 = Ш             217 = Щ             218 = Ъ             219 = Ы
220 = Ь             221 = Э             222 = Ю             223 = Я
224 = а             225 = б             226 = в             227 = г
228 = д             229 = е             230 = ж             231 = з
232 = и             233 = й             234 = к             235 = л
236 = м             237 = н             238 = о             239 = п
240 = р             241 = с             242 = т             243 = у
244 = ф             245 = х             246 = ц             247 = ч
248 = ш             249 = щ             250 = ъ             251 = ы
252 = ь             253 = э             254 = ю             255 = я
```

Приклад 2. Ввести рядок й вивести ASCII-коди його символів.

Текст програми:

```
include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;

int main()
{
   char s[100];
   setlocale(LC_ALL,".1251");   puts("Введіть рядок: ");
   setlocale(LC_ALL,".OCP");    gets_s(s);
   for (int i=0; i<strlen(s); i++)
     printf("%c = %3i  ",s[i],(int)s[i]);
   getch();
   return 0;
}
```

Результати роботи:

```
Введіть рядок:
0123ZzVvЯяАаБб!
0 =  48  1 =  49  2 =  50  3 =  51  Z =  90  z = 122  V =  86  v = 118  Я = -97
 я = -17  А = -128  а = -96  Б = -127  б = -95  ! =  33
```

Приклад 3. Ввести рядок і вивести замість цифр їх ASCII-коди.

Текст програми:

```
include <iostream>
#include <conio.h>
using namespace std;

int main()
{
  char s[100];
  puts("Input one string: ");
  gets_s(s);
  for (int i=0; i<strlen(s); i++)
    if(isdigit(s[i])) printf(" %i ",int(s[i]));
    else printf("%c",s[i]);
  getch();
  return 0;
}
```

Результати роботи:

```
Input one string:
digits: 0123456789
digits:  48  49  50  51  52  53  54  55  56  57
```

Приклад 4. Ввести рядок й обчислити кількість символів '*' у цьому рядку.

Розв'язок. Кількість зірочок обчислимо у функції `zirka`, яка отримає рядок `s`, а поверне до функції `main` ціле число – кількість зірочок у рядку.

Текст програми:

```
#include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;

int zirka(char* s)
{ int i, k=0;
  for (i=0; s[i]!='\0'; i++)
    if(s[i]=='*') k++;  // Якщо знайдено зірочку, збільшити кількість на 1
  return k;
}
int main()
{ setlocale(LC_ALL,".1251");
  char s[100];  int kol;
  puts("Введіть рядок: ");
  gets_s(s);
  kol = zirka(s);       // Виклик функції
  printf("Кількість символів * у рядку =  %i", kol);
  getch();
  return 0;
}
```

Результати роботи:

```
Введіть рядок:
128**Zz* *Яя* *Її
Кількість символів * у рядку =  6
```

Приклад 5. Ввести рядок і визначити, чи є у ньому знаки пунктуації.

Текст програми:

```
#include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;

int znaki(char* s)
{
  for (int i=0; s[i]!='\0'; i++)
   if(s[i]=='.' || s[i]==',' || s[i]==';' || s[i]=='!' || s[i]=='?' || s[i]=='-')
   // або if(ispunct(s[i])) для латиниці
     return 1; //Якщо зустрічається знак пунктуації, функція поверне значення 1,
  return 0;    //а якщо не зустрілося жодного, поверне значення 0
}
```

```
int main()
{  setlocale(LC_ALL,".1251");
   char s[100];   int res;
   puts("Введіть рядок: ");   gets_s(s);
   res = znaki(s);
   if (res) puts("Розділові знаки присутні");
   else puts("Розділових знаків немає");
   getch();
   return 0;
}
```

Результати роботи:

```
Введіть рядок:
Hello, world!
Розділові знаки присутні
```

```
Введіть рядок:
Несе Галя воду
Розділових знаків немає
```

Приклад 6. Ввести два рядки й визначити, чи є вони однаковими.

Текст програми:

```
#include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;
int perevirka(char* s1, char* s2)
{ int i, n1, n2;
  n1=strlen(s1); n2=strlen(s2);
  if (n1 != n2) return 0; // Якщо довжини рядків не збігаються, повернути 0
  for (i=0; i<n1; i++)           // Якщо знайшлись неоднакові символи,
    if(s1[i] != s2[i]) return 0; // повернути 0
  return 1;      // Якщо неоднакових символів не знайдено, повернути 1
}
int main()
{  char s1[100], s2[100];
   setlocale(LC_ALL,".1251");
   puts("Введіть перший рядок: ");  gets_s(s1);
   puts("Введіть другий рядок: ");  gets_s(s2);
   int res = perevirka(s1, s2);
   if(res) puts("Рядки однакові");
   else puts("Рядки не однакові ");
   getch();
   return 0;
}
```

Результати роботи:

```
Введіть перший рядок:
2+4
Введіть другий рядок:
4+2
Рядки не однакові
```

```
Введіть перший рядок:
Hello
Введіть другий рядок:
Hello
Рядки однакові
```

```
Введіть перший рядок:
Програма
Введіть другий рядок:
програма
Рядки не однакові
```

Приклад 7. Перевести до верхнього регістру (у великі літери) всі голосні літери латиниці рядка та обчислити їхню кількість.

```
include <iostream>
#include <conio.h>
using namespace std;
int lowreg(char* s)
{  int k=0;
   for (int i=0; i<strlen(s); i++)
     if (s[i]=='a'||s[i]=='o'||s[i]=='e'||s[i]=='i'||s[i]=='u')
     { s[i]-=32; k++; }
   return k;
}
int main()
{  char s[100];
   puts("Input one string: "); gets_s(s);
   int k=lowreg(s);
   puts(s);
   cout<<"The number of vowels - "<<k<<endl;
   getch();
   return 0;
}
```

```
Input one string:
Open the windows
OpEn thE wIndOws
The number of vowels - 4
```

Приклад 8. Видалити останнє слово з рядка.

```
include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;
void  DeleteLastWord(char* s)
{ int i=0, ind=0;
  while (s[strlen(s)-1]==' ')  s[strlen(s)-1]='\0'; // видалити кінцеві пробіли
  while (s[i]!='\0')
    if(s[i++] ==' ') ind=i; // ind – номер  останнього пробілу в рядку
  s[ind]='\0';              // записати замість пробілу символ кінця рядка
}
int main()
{ char s[100];
  setlocale(LC_ALL,".1251"); puts ("Введіть рядок: ");
  gets_s(s);
  DeleteLastWord(s);
  puts("\nРядок з видаленим останнім словом:");
  setlocale(LC_ALL,".OCP");
  puts(s);
  getch();
  return 0;
}
```

```
Введіть рядок:
Малеча зростає швидко

Рядок з видаленим останнім словом:
Малеча зростає
```

Приклад 9. Ввести два рядки, визначити і вивести всі спільні для обох рядків символи та обчислити їхню кількість.

Текст програми:

```
include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;

int  SameLetters(char* s1, char* s2, char* rez)
{
  rez[0]='\0';
  int i, j, k, m, kol=0;
  for (i=0; s1[i]!='\0'; i++)
    for (j=0; s2[j]!='\0'; j++)
      if (s1[i] == s2[j])
       { for (k=m=0; rez[k]!='\0'; k++)
           if (s1[i]==rez[k])m=1;
         if (!m) { rez[kol]=s1[i]; rez[kol+1]='\0'; kol++;}
       }
return kol;
}

int main()
{
  char s1[100], s2[100],s3[100];
  setlocale(LC_ALL,".1251");    puts("Введіть перший рядок: ");
  setlocale(LC_ALL,".OCP");     gets_s(s1);
  setlocale(LC_ALL,".1251");    puts("Введіть другий рядок: ");
  setlocale(LC_ALL,".OCP");     gets_s(s2);
  int k = SameLetters(s1, s2, s3);
  setlocale(LC_ALL,".1251");
  if (!k) puts("\nНемає однакових символів");
  else printf("\nОднакових символів - %i\n", k);
  setlocale(LC_ALL,".OCP");
  puts(s3);
  getch();
  return 0;
}
```

Результати роботи:

```
Введіть перший рядок:
мама мила раму
Введіть другий рядок:
зараз рама чиста

Однакових символів - 5
ма ир
```

```
Введіть перший рядок:
programmer
Введіть другий рядок:
PROGRAMMER

Немає однакових символів
```

Приклад 10. Вивести ті літери латиниці (і малі, і великі), яких немає у рядку.

Текст програми:

```
include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;

int  NotLetters(char* s, char *rez)
{
  char c;   int i,j,k,kol=0;
  for (c='A',i=0; c<='Z'; c++) rez[i++]=c;
  for (c='a'; c<='z'; c++) rez[i++]=c;
  rez[i]='\0';
  for (i=0; s[i]!='\0'; i++)
    if (s[i]>='A' && s[i]<='Z' || s[i]>='a' && s[i]<='z' )
      for (j=0; rez[j]!='\0'; j++)
        if (s[i]==rez[j])
          { kol++;
            for (k=j; rez[k]!='\0'; k++)  rez[k]=rez[k+1];
          }
  return kol;
}

int main()
{
  char s1[100], s2[55];
  setlocale(LC_ALL,".1251");
  puts("Введіть рядок: ");     gets_s(s1);
  int k = NotLetters(s1,s2);
  if (!k) puts("\nУ рядку немає літер латиниці:");
  else printf("\nУ рядку %i літер(и) латиниці (без урахування регістру),
              \nале серед них немає таких:\n", k);
  puts(s2);
  getch();
  return 0;
}
```

Результати роботи:

```
Введіть рядок:
Голосні літери латиниці такі: aoueiAOUEI

У рядку 10 літер(и) латиниці (без урахування регістру),
але серед них немає таких:
BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz
```

```
Введіть рядок:
Ходить котик тихо, буде мишкам лихо...

У рядку немає літер латиниці:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

Приклад 11. Ввести рядок і вивести окремо у стовпець усі слова цього рядка.

Текст програми:

```
include <iostream>
#include <clocale>
#include <conio.h>
using namespace std;
void words(char* st)
{  char sl[100];      int k=0, i, n;
   strcat(st," ");
   n=strlen(st);
   if (n<2) return;
   sl[0]='\0';
   for (i=0; i<n; i++)
     if (st[i] != ' ')
     {
       sl[k]=st[i];
       sl[k+1]='\0';
       k++;
     }
     else
     {
     if (strlen(sl)>0) puts(sl);
     sl[0]='\0';
     k=0;
   }
}
int main()
{
   char st[100];
   setlocale(LC_ALL,".1251");
   puts("Введіть рядок: ");
   setlocale(LC_ALL,".OCP");
   gets_s(st);
   words(st);
   getch();
   return 0;
}
```

Результати роботи:

```
Введіть рядок:
A NULL pointer indicates an error
A
NULL
pointer
indicates
an
error
```

```
Введіть рядок:
Мама мила раму. Тепер рама чиста.
Мама
мила
раму.
Тепер
рама
чиста.
```

Питання для самоконтролю

1) Дайте означення поняття ASCII. Яка є кількість ASCII-кодів?
2) Скільки байтів виділяється для змінної типу `char`?
3) Які операції можна виконувати над символами?
4) Назвіть вирази, які матимуть значення 1 (`true` – істина – "так"):
   а) `'0' < 'y'`
   б) `'5' > 'f'`
   в) `'F' > 'f'`
   г) `'w' > 'W'`
   д) `' ' > '0'`
   е) `'я' > 'ю'`
5) Як у програмі змінити регістр символів латиниці?

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою С++ програмний код функції та основної програми для розв'язання індивідуальних завдань з опрацювання символьних даних (завдання вибрати з табл. 17.1 … 17.2).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 17.1

Індивідуальні завдання середнього рівня складності

№ вар.	Завдання
1	Визначити кількість малих літер у введеному рядку
2	Замінити великі літери у введеному рядку на пробіли
3	Визначити кількість цифр у введеному рядку
4	Замінити латинські літери у введеному рядку на знаки оклику
5	Замінити розділові знаки у введеному рядку на символи '#'
6	Визначити кількість голосних літер латиниці у введеному рядку
7	Вивести замість літер введеного рядка їхні ASCII-коди
8	Змінити регістр малих латинських літер у введеному рядку
9	Обчислити кількість великих літер у введеному рядку
10	Замінити у введеному рядку цифри на символи '+'
11	Обчислити кількість латинських літер у введеному рядку
12	Визначити індекс першої цифри у введеному рядку
13	Обчислити кількість розділових знаків у введеному рядку
14	Вивести літери латиниці з парними ASCII-кодами та обчислити їх кількість
15	Вивести замість малих літер введеного рядка їхні ASCII-коди
16	Визначити індекс останнього пробілу в рядку
17	Обчислити кількість великих латинських літер у введеному рядку
18	Змінити регістр великих латинських літер у введеному рядку
19	Вивести замість пробілів введеного рядка його ASCII-код
20	Замінити малі латинські літери на крапки у введеному рядку

Закінчення табл. 17.1

№ вар.	Завдання
21	Вивести літери латиниці з непарними ASCII-кодами, обчислити їх кількість
22	Замінити великі латинські літери у введеному рядку на символ коми
23	Вивести замість розділових знаків введеного рядка їхні ASCII-коди
24	Замінити всі пробіли у введеному рядку на знаки підкреслення
25	Визначити, чого більше – ком чи крапок – у рядку, а може їхня кількість однакова?
26	Вивести замість великих літер введеного рядка їхні ASCII-коди
27	У введеному рядку замінити літери латиниці на символ '*'
28	Визначити індекс першого пробілу у введеному рядку
29	Замінити у введеному рядку голосні літери латиниці на символ '%'
30	Визначити, чого у рядку більше – голосних літер латиниці чи цифр, а може їх кількість однакова?

Таблиця 17.2

Індивідуальні завдання підвищеного рівня складності

№ вар.	Завдання
1	Вивести ті слова рядка, довжина яких більше 7-ми
2	Вивести ті слова введеного рядка, які починаються з літери К
3	Знайти найдовшу послідовність літер А, які йдуть у введеному рядку підряд, і визначити її довжину
4	Ввести рядок, який містить круглі дужки. Вивести на екран усі символи, які розміщені усередині цих дужок
5	Визначити, скільки разів у рядку зустрічається комбінація символів "С++"
6	Вивести усі слова, які починаються і закінчуються на одну й ту саму літеру без урахування регістру
7	Визначити, чи є у рядку комбінація символів "С++", і якщо так, вивести індекс її першого входження у рядок
8	Переставити символи рядка у зворотному порядку й обчислити кількість символів, від перестановки яких рядок не змінився
9	Визначити, чи читається рядок однаково зліва направо і справа наліво з урахуванням регістру. Якщо ні, вивести рядок у зворотному порядку
10	Визначити, чи містить рядок лише малі латинські літери. Якщо ні, вивести усі символи, які не є малими латинськими літерами, та їхні індекси
11	Визначити найбільшу кількість пробілів, які йдуть підряд
12	Вивести найдовшу з послідовностей цифр, що є у рядку
13	Змінити регістр малих літер латиниці та обчислити їхню кількість
14	Введіть рядок, всі слова якого зашифровані (записані у зворотному порядку). Розшифрувати текст
15	Видалити з рядка перше слово

Закінчення табл. 17.2

№ вар.	Завдання
16	Видалити з рядка всі пробіли
17	Обчислити кількість слів, які починаються й закінчуються на один і той самий символ з урахуванням регістру
18	Вивести символи першого рядка, яких немає у другому рядку
19	Обчислити кількість слів, які починаються з великої літери
20	Вивести ті великі латинські літери, яких немає у рядку
21	Вивести ті слова рядка, які містять дефіс
22	Визначити і вивести найдовше слово у рядку
23	Визначити середню довжину слів у рядку
24	Вивести слова рядка, довжина яких менше 6, та їх порядкові номери
25	Вивести знаки пунктуації, яких немає у рядку
26	Перевірити, чи є введений рядок послідовністю двійкових цифр (0 і 1). Якщо так, перевести його у десяткову систему. Якщо ні, вивести індекс першого "недвійкового" символу
27	Вивести цифри, яких немає у рядку
28	Вивести маленькі латинські символи, яких немає у рядку
29	Видалити всі зайві (подвійні, початкові та кінцеві) пробіли у рядку
30	Визначити і вивести найкоротше слово у рядку

Лабораторна робота № 18
Рядки char*

Мета роботи: набути практичних навиків програмного опрацювання рядків типу char*.

Теоретичні відомості

Оголошення С-рядків char*

С-рядок може бути оголошеним в один з нижче наведених **способів**:

```
1) char* s;            // Оголошення вказівника на перший символ рядка;
                       // пам'ять під сам рядок не виділяється.
2) char ss[15];        // Оголошення рядка  ss з 14 символів;
                       // пам'ять виділяється компілятором.
3) const int n = 10;
   char st[n];         // Оголошення рядка st з n-1 (тобто 9) символів;
                       // пам'ять виділяється компілятором.
4) int n = 10;
   char* str = new char[n];  // Оголошення рядка str з n-1 (тобто 9)
                             // символів; пам'ять виділяється динамічно.
```

При зазначенні довжини рядка слід враховувати завершальний нуль-символ. Наприклад, у вищенаведеному рядку str варто зберігати не 10, а лише 9 символів.

Зауважимо, що при оголошенні рядка першим способом пам'ять під рядок не виділяється і це може бути дуже небезпечним, оскільки у ту ж саму пам'ять можуть бути розміщені інші змінні і рядок буде втрачено.

При оголошенні рядок можна ініціювати рядковою константою, при цьому нуль-символ формується автоматично після останнього символу:

```
char str[10] = "Vasia";
```

Крім функцій, зазначених у попередній роботі, для введення рядків можна використовувати також cin.getline(). Наприклад:

```
char s[40];
cout<<"Введіть рядок: "<<endl;
cin.getline(s,40);
```

Функції для роботи з рядками char*

C++ має багату колекцію функцій опрацювання рядків char*, найпоширеніші з яких наведено далі в таблиці. Всі ці функції зі стандартної бібліотеки <string.h>. Деякі з цих функцій, наприклад: strlen(), strcpy(), опрацьовують рядки, оголошені в будь-який з чотирьох раніш розглянутих способів. Але більшість функцій потребує, щоб рядок був оголошений як вказівник на тип char*.

Функція	Призначення	Формат
strlen()	повертає довжину рядка (без урахування символу закінчення рядка)	size_t strlen(char *s);
strcat()	приєднує s2 до s1 і, як результат, повертає s1	char *strcat(char *s1,char *s2);

Функ-ція	Призначення	Формат
strncat()	приєднує до рядка s1 перші n символів з рядка s2	char *strncat(char *s1, char *s2, size_t n);
strcmp()	порівнює рядки і повертає нульове значення, (якщо рядки однакові s1=s2), від'ємне (s1<s2) чи додатне (s1>s2). Порівняння відбувається посимвольно і в якості результату повертається різниця кодів перших неоднакових символів	int *strcmp(char *s1, char *s2);
strncmp()	порівнює лише перші n символів рядка s1 з n символами рядка s2	int *strncmp(char *s1, char *s2, size_t n);
stricmp()	порівнює два рядки, не розрізняючи прописні й малі літери латиниці	int stricmp(const char *s1, const char *s2);
strnicmp()	порівнює лише перші n символів двох рядків, не розрізняючи прописні й малі літери латиниці	int strnicmp(const char *s1, const char *s2, size_t maxlen)
strcpy()	копіює в s1 рядок s2, повертає s1, при цьому старе значення s1 втрачається	char *strcpy(char *s1, char *s2);
strncpy()	замінюються перші n символів рядка s1 першими n символами рядка s2	char *strncpy(char *s1, char *s2, size_t n);
strchr()	відшукує в рядку перше входження символу ch і повертає вказівник на цей символ, тобто частину рядка s, починаючи з символу ch і до кінця рядка. Якщо символу в рядку немає, результат – NULL	char *strchr(char *s, char ch);
strrchr()	шукає в рядку s останнє входження символу ch і повертає частину рядка s, починаючи з останнього входження символу ch і до кінця рядка	char *strrchr(char *s, int ch);
strstr()	шукає підрядок s2 в рядку s1, повертає частину рядка s1, починаючи з першого спільного символу для обох рядків і до кінця s1	char *strstr(char *s1, char *s2);
strspn()	повертає довжину початкового сегмента рядка s1, символи якого є в рядку s2	size_t strcspn(char *s1, char *s2);
strcspn()	повертає індекс першого символу в рядку s1, який є спільним для обох рядків (нумерація індексів символів з нуля)	size_t strcspn(char *s1, char *s2);
strlwr()	перетворює латинські літери до нижнього регістру	char *strlwr(char *s);
strupr()	перетворює латинські літери до верхнього регістру	char *strupr(char *s);
strset()	замінює усі символи рядка s символом ch	char *strset(char *s, char ch);
strnset()	замінює лише перші n символів рядка s символом ch	char *strnset(char *s, char ch, size_t n);
strpbrk()	знаходить місце першого входження в рядок s1 будь-якого символу рядка s2 і повертає частину рядка s1, починаючи з цього символу і до кінця	char *strpbrk(char *s1, const char *s2);
strrev()	реверс рядка s	char *strrev(char *s);
strtok()	повертає частину (лексему) рядка s1 від поточної позиції вказівника до розділового символу, зазначеного у рядку s2; звичайно використовується для перетворення рядка у послідовність лексем	char *strtok(char *s1, const char *s2);

Приклади програм

Приклад 1. Ввести рядок й визначити довжину першого й останнього слів у рядку (розділювачем слів вважати пробіл).

Розв'язок. Для визначення довжини першого слова слід знайти індекс першого пробілу. Для визначення довжини останнього слова треба знайти індекс останнього пробілу, обчислити різницю довжини рядка та цього індексу і зменшити її на 1.

Текст програми:

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
#include <clocale>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 char *s=new char [100];
 puts(" Введіть рядок:");
 gets_s(s, 100);
 int pp=0, op=0, n=strlen(s), i;
 for (i=0; i<n; i++)
   if (s[i]==' ')   // Зупинитись, якщо зустріли перший пробіл
     { pp=i;  cout << "\n Довжина першого слова: " << pp << endl;  break;}
 if(!pp) {cout<<"\nПробілів немає"<<endl; cin.get();  return 0;}
 for (i=n-1; i>=0; i--)
   if (s[i]==' ') { op=i;  break;} // Зупинитись, якщо зустріли пробіл
 cout<< "\n Довжина останнього слова:  " << n-op-1 << endl;
 delete []s;
 cin.get();
 return 0;
}
```

Така само програма зі створенням функцій:

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
#include <clocale>
using namespace std;

int first_space(char *s) // Пошук першого пробілу
{ int p=-1;
  for (int i=0; s[i]!='\0'; i++)
    if (s[i]==' ') {p=i; break;}
  return p;
}
```

```
int last_space(char *s) // Пошук останнього пробілу
{ int p=-1;
  for (int i=0; s[i]!='\0'; i++)
    if (s[i]==' ') p=i;
  return p;
}

int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 int pp, op, n;
 char *s=new char [100];
 puts(" Введіть рядок:");
 gets_s(s, 100);
 n=strlen(s);
 pp=first_space(s);
 if (pp>-1)
   cout<< "\n Довжина першого слова:  " << pp << endl;
 else
   {  cout<<"\nПробілів немає" << endl;  cin.get();  return 0;}
 op=last_space(s);
 cout<< "\n Довжина останнього слова:  " << n-op-1 << endl;
 delete [] s;
 cin.get();
 return 0;
}
```

Результат виконання програми:

```
 Введіть рядок:
програма

Пробілів немає
```

```
 Введіть рядок:
Одеська національна академія зв'язку - super

 Довжина першого слова:  7
```

Приклад 2. На основі введеного рядка створити новий рядок з маленьких голосних літер латиниці, які містяться у введеному рядку.

Розв'язок. Спочатку треба обчислити кількість маленьких голосних літер латиниці у введеному рядку `s`, щоб виділити необхідну кількість пам'яті для символів створюваного рядка `s1`. Після цього можна копіювати у новий рядок `s1` голосні маленькі літери латиниці з рядка `s`, а наприкінці долучити завершальний нуль до нового рядка.

Текст програми:

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
#include <clocale>
```

```
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 int i, n, kol=0, j=0;
 char *s=new char [100];
 puts(" Введіть рядок:");
 gets_s(s, 100);
 n=strlen(s);
 for (i=0; i<n; i++)
   if (s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u') kol++;
 char *s1 = new char [kol+1];
 for (i=0; i<n; i++)
   if (s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u')
     { s1[j]=s[i]; j++; }
 s1[kol]='\0';
 puts("\n Новий рядок з голосних літер  ");
 setlocale(0, ".OCP");
 puts (s1);
 delete [] s;
 cin.get();
 return 0;
}
```

Така ж сама програма зі створенням функцій:

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
#include <clocale>
using namespace std;

int kol_vo(char *s)  // Визначення кількості голосних маленьких літер
{ int k=0, i;
  for (i=0; s[i]!='\0'; i++)
   if (s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u') k++;
  return k;
}
void new_string(char *s, char *s1) // Заповнення нового рядка
{ int i, j=0;
  for (i=0; s[i]!='\0'; i++)
      if (s[i]=='a' || s[i]=='e' || s[i]=='i' || s[i]=='o' || s[i]=='u')
        {s1[j]=s[i]; j++;}
   s1[j]='\0';
}
int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 char *s = new char [100];
```

```
 puts(" Введіть рядок:");
 gets_s(s, 100);
 int kol = kol_vo(s);
 char *s1 = new char [kol+1];
 new_string(s, s1);
 puts("\n Новий рядок з голосних літер  ");
 setlocale(0, ".OCP");
 puts(s1);
 delete []s;
 cin.get();
 return 0;
}
```

Результат виконання програми:

```
 Введіть рядок:
Visual Studio Integrated Development Environment

 Новий рядок з голосних літер
iuauioeaeeeoeioe
```

Приклад 3. Ввести рядок і вивести всі заголовні літери, які стоять на початку слів. Слова розділяються одним чи кількома пробілами.

Текст програми:

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
#include <clocale>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 int i, n, kol=0, j=0;
 char *s=new char [100];
 puts(" Введіть рядок:");
 gets_s(s, 100);
 n=strlen(s);
 puts(" Результат:");
 setlocale(0, ".OCP");
 if (s[0]!=' ' && isupper(s[0])) // Перше слово перевіряємо окремо
      cout<<s[0]<<endl;
 for (i=0; i<n-1; i++)
   if (s[i]==' ' && s[i+1]!=' ' && isupper(s[i+1]))
      cout<<s[i+1]<<endl;
 delete [] s;
 cin.get();
 return 0;
}
```

```
 Введіть рядок:
Development Tools and Language
 Результат:
D
T
L
```

Приклад 4. Ввести рядок і вивести всі його слова окремо без розділових знаків.

Розв'язок. Для розбивання рядка на слова (лексеми) можна застосувати функцію `strtok()`, яка має два аргументи: 1) рядок, який треба розбити на лексеми, і 2) рядок, який містить символи-розділювачі.

Перший виклик функції `strtok()` у нижченаведеній програмі

```
t = strtok(s, " .,;?!-");
```

присвоює `t` вказівник на першу лексему (слово) в рядку `s`. Другий аргумент містить можливі розділові символи. Функція `strtok()` шукає перший символ у рядку `s`, який не є розділювачем. Це початок першого слова. Потім функція знаходить наступний розділювач у рядку і замінює його нуль-символом (`'\0'`). Цим закінчується поточне слово. Функція `strtok()` зберігає вказівник на наступний символ рядку `s` за даною лексемою (словом) і повертає вказівник на поточну лексему (слово).

У наступних викликах `strtok()` у програмі для почергового розбивання рядка `s` на слова першим аргументом функції є `NULL` для того, щоб виокремлення наступного слова рядка `s`, відбувалося з позиції, яку було збережено попереднім викликом `strtok()`. Якщо лексем при виклику `strtok()` більше немає, `strtok()` повертає `NULL` і відбувається вихід із циклу.

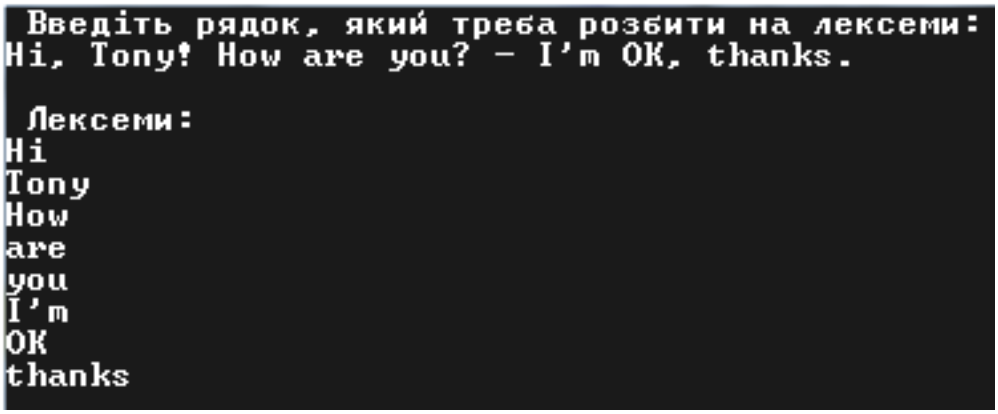
Зверніть увагу, що `strtok()` модифікує вхідний рядок; тому, якщо рядок після виклику `strtok()` буде знову використовуватись у програмі, слід завчасно зробити копію рядка.

Цей алгоритм можна застосовувати для виокремлення чисел, але тоді у якості розділових знаків не слід застосовувати крапку, яка може інтерпретуватися як десяткова крапка.

Текст програми у консолі:

```
#include "stdafx.h"
#include <iostream>
int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 char *s = new char [80], *t;
 puts(" Введіть рядок, який треба розбити на лексеми:"); gets_s(s, 80);
 puts("\n Лексеми:  ");
 setlocale(0, ".OCP");
 t = strtok(s, " .,;?!-");
 while (t != NULL)
 { puts(t);
   t = strtok(NULL, " .,;?!-");
 }
 delete []s;
 system("pause>>void");
 return 0;
}
```

Результат виконання програми:



Приклад 5. Ввести рядок і видалили з нього друге слово.

Розв'язок. У нижченаведеному коді розв'язання організовано в окремій функції `del_word()`, яка видаляє з рядка `s`, починаючи з позиції `k`, `n` символів. У цій функції спочатку копіюється з рядка `s` перше слово з пробілом у допоміжний рядок `s1`. Після чого долучається до `s1` решта рядка `s` без слова, яке має бути видаленим (адреса початку цієї частини рядка обчислюється за допомогою адресної арифметики – зсув на `k+n+1` символів від початку рядка `s`).

Текст програми

```
#include "stdafx.h"
#include <iostream>
#include <string.h>
using namespace std;

void del_word(char *s, int k, int n)
{ char *s1=new char [strlen(s)+1];
  strncpy(s1, s, k);  s1[k]='\0';
  strcat(s1, s+k+n+1);
  strcpy(s,s1);
  delete [] s1;
}

int _tmain(int argc, _TCHAR* argv[])
{
 setlocale(LC_ALL,".1251");
 int i, kol=0, p1=-1, p2=-1;
 char *s=new char [100];
 puts(" Введіть рядок:"); gets_s(s, 100);
 strcat(s," ");
 for (i=0; s[i]!='\0'; i++)
  if (s[i]==' ')
   if (p1==-1) p1=i;
   else { p2=i; break; }
 del_word(s, p1+1, p2-p1-1);
 puts("\n Рядок без другого слова ");
 setlocale(0, ".OCP");  puts(s);
```

```
 delete []s;
 system("pause>>void");
 return 0;
}
```

Результат виконання програми:

```
 Введіть рядок:
Microsoft developer network library

 Рядок без другого слова
Microsoft network library
```

```
 Введіть рядок:
Моя рідна країна

 Рядок без другого слова
Моя країна
```

Приклад 6. Ввести рядок і видалили з нього слова, коротші 4 символів.

Розв'язок. Для розв'язання задачі створимо новий рядок `s1` зі слів рядка `s`, довжина яких не менше 4-х символів.

Текст програми

```
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{ setlocale(LC_ALL,".1251");
  char* s=new char [80], *t;
  char* s1=new char [80];
  strcpy(s1,"\0");
  puts(" Введіть рядок:");  gets_s(s, 80);
  t=strtok(s, " .,;?!-");
  while (t != NULL)
    { if (strlen(t)>=4) { strcat(s1,t); strcat(s1, " \0"); }
      t = strtok(NULL, " .,;?!-");
    }
 puts("\n Рядок без коротких слів ");
 setlocale(0, ".OCP");
 strcpy(s, s1);
 puts(s);
 delete [] s;
 delete [] s1;
 system("pause>>void");
 return 0;
}
```

```
 Введіть рядок:
Microsoft SQL Server is a database management

 Рядок без коротких слів
Microsoft Server database management
```

Приклад 7. Для введенего рядка поміняти місцями вказані номерами слова.

Текст програми

```
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{ setlocale(LC_ALL,".1251");
```

```
  char s[100] = {'\0'}, rez[100] = {'\0'},*word;
  char swap[25], a[25][20];     // для переставляння слів
  int k = -1, i, l, r;
  printf("Введіть  рядок:\n");
  gets(s);
  word = strtok(s, " \'\":-!,");
  while(word)
  {  k++;                       // кількість слів
     strcpy(a[k], word);
     word = strtok(NULL, " \'\":-!,");
  }
  printf("Всього %d слів(ова)\n" ,k+1);
  printf("\nВведіть номери слів у рядку для обміну: ");
  if(scanf("%d %d", &l, &r)==0 || l>r || r>k+1)
  {  printf("\nНеправильне введення номерів слів для обміну!!!\n");
     return 1;
  }
  strcpy(swap,a[l-1]);          // переставляння слів у масиві слів
  strcpy(a[l-1],a[r-1]);
  strcpy(a[r-1],swap);
  for(i=0;i<=k;i++)
  {  strcat(rez,a[i]);          // поєднання слів у рядок
     strcat(rez," ");
  }
  puts("\nПісля обміну слів:");
  setlocale(LC_ALL,".OCP");
  puts(rez);
  system("pause>>void");
  return 0;
}
```

```
Введіть  рядок:
Нехай все буде добре
Всього 4 слів(ова)

Введіть номери слів у рядку для обміну: 3 4


Після обміну слів:
Нехай все добре буде
```

Питання для самоконтролю

1) Наведіть відомі способи оголошення С-рядка.
2) Чи можна присвоїти С-рядку частину іншого рядка? Якщо так, наведіть приклад.
3) Яка функція дозволяє з'єднати два С-рядки? Наведіть приклад.
4) Наведіть способи введення С-рядка у консолі.

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою С++ програми для розв'язання індивідуальних завдань з використанням стандартних функцій для опрацювання рядків (завдання вибрати з табл. 18.1 … 18.2).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 18.1

Індивідуальні завдання базового рівня складності

№ вар.	Завдання
1	Вивести перше слово введеного рядка
2	Знайти індекси початку й кінця другого слова у введеному рядку
3	Вивести перше слово введеного рядка у зворотному порядку
4	Вивести останнє слово введеного рядка
5	Визначити кількість слів у введеному рядку. Слова розділяються одним чи декількома пробілами
6	Для двох введених рядків побудувати третій, на початку якого стоятиме довший з рядків, а за ним через пробіл – коротший
7	Створити новий рядок з першого й останнього слів введеного рядка
8	Вивести перші літери всіх слів введеного рядка. Слова розділяються одним чи декількома пробілами
9	Створити новий рядок з цифр, які містяться у введеному рядку
10	Вивести індекси всіх входжень літери S до рядка
11	Визначити кількість слів рядка, які починаються з літер латиниці
12	Вести рядок з літер латиниці і виконати над ним такі перетворення регістру: якщо перша літера рядка є великою (верхній регістр) перевести до верхнього регістру весь рядок , і навпаки, якщо перша літера мала – до нижнього
13	Вивести останні літери усіх слів введеного рядка. Слова розділяються одним чи декількома пробілами
14	Вивести індекси всіх цифр, які є у введеному рядку
15	Вивести всі розділові знаки та їхні індекси у введеному рядку
16	Перевернути у зворотному порядку останнє зі слів введеного рядка
17	Перевести перше зі слів введеного латиницею рядка до верхнього регістру
18	Вивести усі маленькі латинські літери, які стоять на початку слів. Слова розділяються одним чи декількома пробілами
19	Вивести індекси всіх ком та крапок, які є у введеному рядку
20	Визначити кількість зайвих пробілів у рядку (подвійних і початкових)
21	Створити новий рядок з великих літер латиниці введеного рядка
22	Визначити кількість слів, після яких є розділові знаки, та вивести ці знаки
23	Визначити, чи є у рядку знаки арифметичних дій. Якщо так, то вивести на екран ці знаки
24	Перевернути у зворотному порядку перше зі слів введеного рядка
25	Видалити усі пробіли у введеному рядку
26	Створити новий рядок з першого та другого слів введеного рядка
27	Створити новий рядок з малих літер латиниці введеного рядка
28	Створити новий рядок з розділових знаків введеного рядка
29	Визначити, чи є у введеному рядку знаки арифметичних дій. Якщо так, то створити новий рядок з цих знаків
30	Перетворити останнє слово введеного латиницею рядка до нижнього регістру

Таблиця 18.2

Індивідуальні завдання підвищеного рівня складності

№ вар.	Завдання
1	Видалити ті слова рядка, які мають цифри
2	Вивести найдовше слово введеного рядка у зворотному порядку
3	Створити новий рядок зі слів введеного рядка, довжина яких більше 6-ти
4	Вивести на екран ті слова введеного рядка, які починаються й закінчуються на один і той самий символ
5	Видалити передостаннє слово введеного рядка
6	Вивести найкоротше слово введеного рядка
7	Ввести рядок та окремо слово, яке є в цьому рядку. Видалити з рядка перше входження цього слова
8	Видалити всі слова введеного рядка, які починаються на літеру А
9	Створити рядок зі слів введеного рядка, довжина яких менше 5-ти символів
10	Видалити з рядка слова, які починаються і закінчуються однаковою літерою
11	Для введеного рядка вставити після кожного слова знак оклику
12	Видалити ті слова рядка, які закінчуються цифрою
13	Видалити з рядка усі зайві пробіли (подвійні, початкові і кінцеві)
14	Переставити у введеному рядку перше й останнє слова
15	Для двох введених рядків поміняти місцями їхні перші слова
16	У введеному рядку видалити всі розділові символи
17	Створити новий рядок з тих слів введеного рядка, які починаються з малих літер латиниці
18	Переставити у введеному рядку перше і друге слова
19	У введеному рядку видалити всі символи, які не є цифрами та крапкою
20	Вставити перед та після кожного слова введеного рядка символ ‘*’
21	У введеному рядку видалити слова з парними номерами
22	Створити новий рядок зі слів введеного рядка, які починаються з великих літер латиниці
23	Створити новий рядок зі слів введеного рядка з непарними номерами
24	Вивести стовпчиком у зворотному порядку всі слова введеного рядка
25	Створити новий рядок з найкоротшого й найдовшого слів введеного рядка
26	Видалити з рядка найкоротше слово
27	Вивести ті слова введеного рядка, які не містять цифр
28	Ввести рядок і окремо слово. Вставити це слово після першого слова рядка
29	Створити новий рядок зі слів введеного рядка, які починаються з літери К
30	Видалити з рядка найдовше слово

Лабораторна робота № 19

Рядки класу String

Мета роботи: набути практичних навиків програмного опрацювання рядків класу `String`.

Теоретичні відомості

Оголошення та специфіка організації рядків класу String

Рядки класу `String` представляють текст як послідовність символів Unicode і належать простору імен `System`. Оголошуються вони так:

```
String ^<змінна>;
```

Наприклад:

```
String ^s, ^str = "abcdefg";
```

При такому оголошенні друга зі змінних не лише оголошується, а й ініціалізується початковим значенням. Ініціалізація оголошуваного рядка значенням, введеним з клавіатури, може мати вигляд:

```
String ^myString = Console::ReadLine();
```

Для визначення довжини рядка можна скористатися властивістю `Length`:

```
int n = str->Length;
```

Максимальний розмір об'єкта `String` – 2 ГБ пам'яті, або близько 1 млрд. символів.

Рядки класу `String` складаються з символів типу `Char`, тобто символів Unicode з кодуванням UTF-16 (16-бітове значення з діапазону шістнадцяткових значень 0x0000 – 0xFFFF).

Звернення до кожного символу рядка виконується за допомогою індексу, як до елемента масиву:

```
myString [i]
```

Конкатенацію (зчеплення) рядків виконують за допомогою оператора `+`:

```
String ^s = "This is one sentence. " + "This is a second. ";
s += "This is a third sentence.";
Console::WriteLine(s);
```

У результаті цього сформується і виведеться на екран рядок:

```
"This is one sentence. This is a second. This is a third sentence."
```

Деякі методи для роботи з рядками

1) **`IndexOf` – пошук підрядка у рядку**. Метод повертає індекс (нумерація від нуля) першого входження підрядка або зазначеного символу Unicode в межах даного екземпляра. Метод повертає -1, якщо знак або підрядок не знайдено в даному екземплярі. Пошук виконується з місця, вказаного у другому параметрі. Якщо другий параметр відсутній, то пошук розпочинається з початку рядка.

Наприклад, команда для визначення номера першого пробілу в рядку `s` матиме вигляд:

```
int n = s->IndexOf(" ");
```

Метод `IndexOf`, як і більшість методів для роботи з класом `String`, мають декілька форм: з одним, двома і, навіть, трьома аргументами. З докладнішою інформацією можна ознайомитись за Інтернет-адресою:

http://msdn.microsoft.com/ru-ru/library/system.string.indexof(v=vs.110).aspx.

2) **`Substring` – отримання підрядка з рядка**, тобто копіювання частини рядка, починаючи з певної позиції і до кінця рядка:

```
String^ Substring(int startIndex)
```

Друга форма цього методу дозволяє задавати певну довжину копійованої частини рядка:

```
String^ Substring(int startIndex, int length)
```

Наприклад, виокремлення другого слова рядка `s` можна здійснити так:

```
String ^s = "This sentence has five words." , ^word2;
int startPos = s->IndexOf(" ") + 1;
word2 = s->Substring(startPos, s->IndexOf(" ",startPos) – startPos);
```

У цьому фрагменті:

`s->IndexOf(" ")+1` – індекс початку другого слова (наступний символ після пробілу);

`IndexOf(" ", startPos)` – індекс пробілу після другого слова;

`s->IndexOf(" ",startPos)–startPos` – довжина другого слова.

3) **`Insert` – вставляння підрядка в задану позицію поточного рядка**.

Наприклад, після виконання команд

```
String ^s = ".NET Framework 4.5";
s = s->Insert(0, "C++");
```

рядок `s` набуде вигляду: `"C++.NET Framework 4.5"`, тобто на початок рядка буде вставлено фрагмент `"C++"`.

4) **`Remove` – видалення з рядка певної кількості символів** із заданої позиції поточного рядка:

```
Remove (int startIndex, int count)
```

Наприклад, наступний набір команд видалить з рядка друге слово – набір символів між першим і другим пробілами, тобто значення по-батькові:

```
String ^name = "Микола Петрович Тараненко";
int foundS1 = name->IndexOf(" ");
int foundS2 = name->IndexOf(" ", foundS1 + 1);
name = name->Remove(foundS1 + 1, foundS2 - foundS1);
```

Якщо другого слова в рядку не буде, видаляться всі символи до кінця рядка.

5) **`Replace` – заміна частини рядка**. Метод замінює всі входження в рядок певної сукупності символів `oldValue` на іншу сукупність символів `newValue`:

```
Replace (String^ oldValue, String^ newValue)
```

Наприклад, змінити слово математика в рядку `s` (якщо воно там є) на нову назву можна командою:

```
s = s->Replace("математика", "вища математика");
```

6) **Методи `ToLower` і `ToUpper`** перетворюють усі символи рядка до нижнього чи то до верхнього регістру, при чому методи коректно опрацьовують і латиницю, і кирилицю. Наприклад:

```
String ^s = Console::ReadLine();
Console::WriteLine(s->ToUpper());
```

7) **Методи `TrimStart`, `TrimEnd` і `Trim`** дозволяють позбавитись або початкових, або кінцевих, або і тих і тих пробілів у рядку.

8) **`Split` – поділ рядка на порції, залежно від вказаного розділювача**. Метод відразу формує масив рядків (`array<String^>`) з виокремленими порціями символів – підрядками, які були розміщені у вихідному рядку між будь з яких зазначених розділювачів, наприклад:

```
String ^s = "Тараненко    12.06.1998    167";
array<Char> ^Сепаратор = {' ', '\t' };
array<String^> ^subs =
          s->Split(Сепаратор, StringSplitOptions::RemoveEmptyEntries);
```

У наведеному прикладі другий параметр в методі `Split` вказує на те, що порожні елементи не будуть включатися до масиву формованих підрядків. Результатом виконання наведеної команди буде масив `subs` з трьох елементів: `subs[0]="Тараненко"`; `subs[1]="12.06.1998"`; `subs[2]=167"`. Розділювачами були вказані пробіл і символ табуляції, причому їхня кількість чи поєднання не мають значення, оскільки вони не включаються до набору елементів формованого масиву.

9) **`Compare` – порівняння двох рядків**. Метод повертає ціле число, яке показує їх відносне положення в порядку сортування: –1 (якщо перший рядок менше другого), 0 (якщо рядки однакові) або 1 (якщо перший рядок більше другого). При порівнянні рядків відбувається почергове посимвольне порівняння кодів символів у рядках. Наприклад, в умовному операторі

```
if (String::Compare("2+1","1+2")>0) . . .
```

умова буде істинною, оскільки код символа '2' є більшим за код символа '1'.

Деякі методи для роботи з символами Char

1) `IsDigit` – перевірка на те, чи є символ десятковою цифрою. Якщо символ є цифрою результат – `true`, інакше – `false`.

Звичайно, метод використовують разом з оператором `if` у такий спосіб:

```
if (Char::IsDigit(s[i])) . . .
```

2) `IsLetter` – повертає значення `true`, якщо символ є будь-якою **літерою** Unicode (кирилиця також), інакше – `false`.

3) `IsLower` – повертає значення `true`, якщо символ є **літерою нижнього регістру** (малою літерою), інакше – `false`.

4) `IsUpper` – повертає значення `true`, якщо символ є **літерою верхнього регістру** (великою літерою), інакше – `false`.

5) `IsPunctuation` – повертає значення `true`, якщо символ є одним зі знаків **пунктуації**: '.', ',', '!', ';', ':', '?', '(', ')', '[', ']' та ін.

6) `IsWhiteSpace` – повертає значення `true`, якщо символ є пробілом, інакше – `false`.

7) `Equals` – повертає значення `true`, якщо символ є однаковим (збігається) з порівнюваним значенням, наприклад:

```
for (int i = 0; i < s->Length; i++)
 if (s[i].Equals('*'))  { s = s->Remove(i, 1); i--;}
```

Ці команди видалять всі зірочки з рядка.

Приклади програм

Приклад 1. Обчислити кількість пробілів і знаків пунктуації у введеному рядку.

Текст програми:

```
using namespace System;
int main(array<System::String ^> ^args)
{
  Console::WriteLine("Введіть рядок:");
  String ^s = Console::ReadLine();
  int nSpace = 0, nPunct=0;
  for (int i = 0; i < s->Length; i++)
  {
    if (Char::IsPunctuation(s[i])) nPunct++;
    if (Char::IsWhiteSpace(s[i]))  nSpace++;
  }
  Console::WriteLine("У введеному рядку\n{0}\nрозміщено {1} пробіл(и/ів)
                      і {2} знак(и/ів) пунктуації ",  s, nSpace, nPunct);
  Console::ReadLine();
  return 0;
}
```

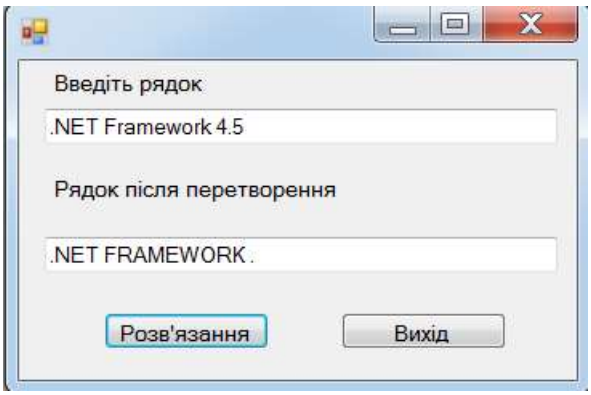
Результат виконання програми:

```
Введіть рядок:
На все добре!
У введеному рядку
На все добре!
розміщено 2 пробіл(и/ів) і 1 знак(и/ів) пунктуації
```

Приклад 2. У введеному рядку спочатку видалити всі цифри і перевести символи рядка до верхнього регістру.

Текст програми:

```
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{
  int kol=0;
  String^ str = textBox1->Text;
  for (int i=str->Length-1; i>=0; i--)
    if (Char::IsDigit(str[i]))
          str = str->Remove(i, 1);
  str = str->ToUpper();
  textBox2->Text = str;
}
```



Приклад 3. Замінити в рядку цифри на їхні словесні назви, наприклад: 1 – на "один", 2 – на "два" тощо.

Розв'язок. Для таких замін доцільно використати метод `Replace()`, але оскільки цей метод може працювати з аргументами – рядками, то спочатку символ слід перетворити до рядка за допомогою метода `ToString()`. Вибір словесної назви тієї чи іншої цифри зручно організувати в операторі `switch`, вказавши в якості ключа сам символ. При цьому, насправді, буде опрацьовуватись код символу.

Текст програми:

```
using namespace System;
int main(array<System::String ^> ^args)
{
String ^s = "2 * 2 = 4 - это всем известно в целом мире,
            \na не 6 и не 5 - это надо знать!";
 Console::WriteLine(s);
 int i=0; String ^nazva;
 for (int i = 0; i < s->Length-1; i++)
  if(Char::IsDigit(s[i]))
  { switch(s[i])
    { case '0': nazva="ноль";break;
      case '1': nazva="один";break;
      case '2': nazva="два";break;
      case '3': nazva="три";break;
      case '4': nazva="четыре";break;
      case '5': nazva="пять";break;
      case '6': nazva="шесть";break;
      case '7': nazva="семь";break;
      case '8': nazva="восемь";break;
      case '9': nazva="девять";break;
     }
   s = s->Replace(s[i].ToString(), nazva);
  }
  Console::WriteLine("\nСтрока после замены цифр на их названия:\n"+s);
  Console::ReadLine();
 return 0;
}
```

Результат виконання програми:

```
2 * 2 = 4 - это всем известно в целом мире,
а не 6 и не 5 - это надо знать!

Строка после замены цифр на их названия:
два * два = четыре - это всем известно в целом мире,
а не шесть и не пять - это надо знать!
```

Приклад 4. У введеному рядку вставити після кожного символу '*' його порядковий номер та номер позиції цього символу у рядку.

Текст програми:

```
using namespace System;
int main(array<System::String ^> ^args)
{
  Console::WriteLine("Введіть рядок:");
  String  ^s = Console::ReadLine();
  int kol = 0, i;
  for (i = 0; i <= s->Length-1; i++)
    if (s[i].Equals ('*'))
    { kol++;
      s = s->Insert(i+1, " (nom="+kol.ToString()+" pos="+i.ToString()+") ");
    }
  Console::WriteLine("\nРезультат:\n"+s);
  Console::ReadLine();
 return 0;
}
```

Результат виконання програми:

```
Введіть рядок:
*qw*33*

Результат:
* (nom=1 pos=0) qw* (nom=2 pos=18) 33* (nom=3 pos=37)
```

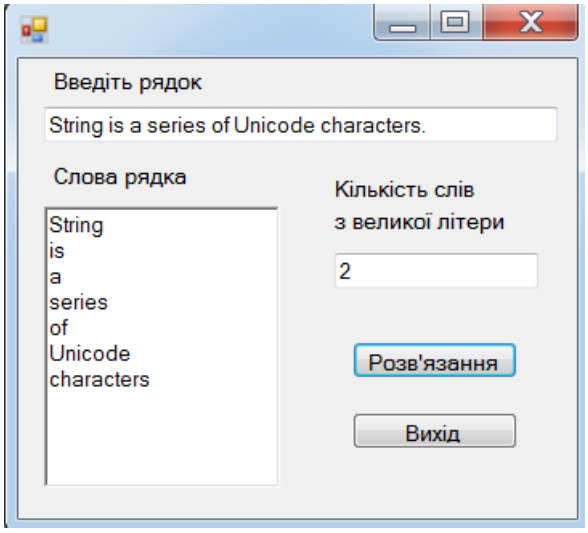
Приклад 5. Ввести рядок, вивести окремо кожне зі слів цього рядка та обчислити кількість слів рядка, які починаються з великої літери.

Розв'язок. Зверніть увагу, що у наведеному програмному коді у методі Split параметр StringSplitOptions::RemoveEmptyEntries використано для виключення порожніх рядків, які можуть виникнути у разі введення подвійних пробілів тощо.

Текст програми:

```
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{   int kol = 0;
    String ^str = textBox1->Text;
    String ^delimStr = " ,.:\t";
    array<Char>^ delimiter = delimStr->ToCharArray( );
    array<String^> ^words;
    words = str->Split(delimiter, StringSplitOptions::RemoveEmptyEntries);
    for (int i=0; i<words->Length; i++)
    {  richTextBox1->Text += words[i]+"\r\n";
       if (Char::IsUpper(words[i][0])) kol++;
    }
    textBox3->Text = Convert::ToString(kol);
}
```

Результат виконання програми:



Приклад 6. У введеному рядку перевести голосні латинські літери до верхнього регистру.

Текст програми:

```
Введіть рядок:
Hello, world!

Результат:
HEllO, wOrld!
```

```
using namespace System;

int main(array<System::String ^> ^args)
{
  Console::WriteLine("Введіть рядок:");
  String ^s = Console::ReadLine();
  for (int i = 0; i < s->Length; i++)
    if (s[i]=='a'||s[i]=='o'||s[i]=='e'||s[i]=='i'||s[i]=='u')
    {  Char c = Char::ToUpper(s[i]);
       s = s->Remove(i,1);
       s = s->Insert(i,c.ToString());
    }
  Console::WriteLine("\nРезультат:\n" + s);
  Console::ReadLine();
  return 0;
}
```

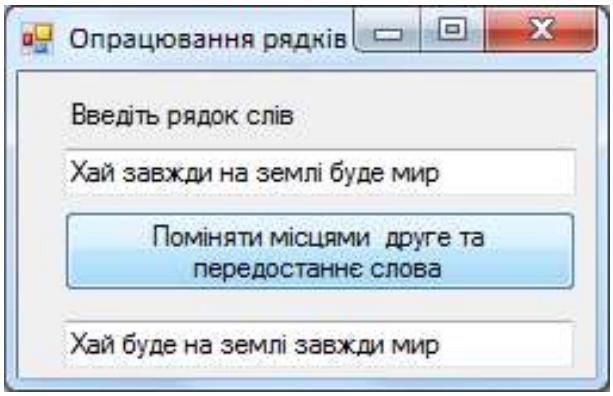
Приклад 7. У введеному рядку поміняти друге та передостаннє слова.

Текст програми:

```
private: System::Void button1_Click(System::Object^sender,System::EventArgs^e)
{ String^ s = textBox1->Text->Trim();
  String ^dS = " .,:!?\t";
  array<Char>^ del = dS->ToCharArray();
  array<String^> ^words = s->Split(del, StringSplitOptions::RemoveEmptyEntries);
  int n = s->IndexOf(words[words->Length-2]); // позиція передостаннього слова
  s = s->Remove(n,words[words->Length-2]->Length); // видалити передостаннє слово
  s = s->Insert(n,words[1]);                  // поставити замість видаленого друге слово
```

```
  n = s->IndexOf(words[1]);                 // позиція другого слова
  s = s->Remove(n,words[1]->Length);        // видалити друге слово
  s = s->Insert(n,words[words->Length-2]); // поставити туди передостаннє слово
  textBox2->Text = s;
}
```

Результат виконання програми:



Питання для самоконтролю

1) Поясніть відмінності символів `char` від `Char` і рядків `char*` від `String`.
2) Наведіть приклад оголошення та введення рядка класу `String`.
3) Що виведеться на екран після виконання команд:

```
String ^s = ".NET Framework 4.5";
Console::WriteLine(s->ToUpper());
```

Скільки символів рядка `s` після цього залишиться незмінними?

4) Що виведеться на екран після виконання команд:

```
String ^s = ".NET Framework 4.5";
s = s->Insert(0, "C++");
Console::WriteLine(s);
```

5) Що виведеться на екран після виконання команд:

```
String ^s = ".NET Framework 4.5";
s = s->Remove(5, 9);
Console::WriteLine(s);
```

6) Запишіть команду, яка дозволить знайти у рядку `s` індекс першого пробілу?

7) Запишіть команду, яка дозволить вставити на початок рядка `s` слово `"good"`?

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою C++ програми для розв'язання індивідуальних завдань з опрацювання рядків класу `String` (завдання вибрати з табл. 19.1 … 19.2).

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм. Занести результати обчислень до протоколу.

Таблиця 19.1

Індивідуальні завдання базового рівня складності

№ вар.	Завдання
1	Визначити кількість цифр у введеному рядку
2	У введеному рядку замінити знаки оклику крапками
3	У введеному рядку перетворити у верхній регістр символи з парними індексами
4	У введеному рядку перетворити у верхній регістр перші літери слів (ті, що йдуть після пробілу)
5	Визначити кількість великих літер у введеному рядку
6	Визначити кількість знаків пунктуації у введеному рядку
7	У введеному рядку замінити всі знаки пунктуації пробілами
8	У введеному рядку замінити подвійні пробіли знаками підкреслення
9	Визначити індекс першої цифри у введеному рядку
10	У введеному рядку визначити символ з найменшим кодом. Вивести цей символ та його номер коду
11	Визначити, чого більше у введеному рядку: цифр або літер
12	У введеному рядку поміняти місцями перший і останній символи
13	Визначити кількість слів у введеному рядку
14	Визначити кількість маленьких літер у введеному рядку
15	У введеному рядку визначити кількість символів, які не є ані літерами, ані цифрами
16	У введеному рядку замінити всі цифри на символ ‘+’
17	Визначити кількість літер кирилиці у введеному рядку
18	Обчислити кількість латинських літер у введеному рядку
19	Визначити індекс першої великої латинської літери у введеному рядку
20	На основі введеного рядка створити новий рядок з літер кирилиці першого
21	У введеному рядку поміняти місцями перший і останній символи першого слова
22	Визначити символ з найбільшим кодом у введеному рядку. Вивести цей символ та його номер коду
23	Визначити індекс останнього пробілу у введеному рядку
24	У введеному рядку перетворити всі символи з непарними індексами на нижній регістр
25	Замінити всі великі літери введеного рядка на символ ‘*’
26	На основі введеного рядка створити новий рядок з символів з парними індексами
27	У введеному рядку визначити, літер якого регістру більше: верхнього або нижнього?
28	Вивести всі символи введеного рядка у зворотному порядку
29	Замінити крапки і коми дефісами у введеному рядку
30	У введеному рядку обчислити кількість символів після першого пробілу

Таблиця 19.2

Індивідуальні завдання підвищеного рівня складності

№ вар.	Завдання
1	Вивести всі слова введеного рядка, які починаються з великої літери
2	Обчислити кількість слів введеного рядка, які містять літеру ‘а’
3	Визначити кількість слів введеного рядка, які починаються й закінчуються на одну й ту саму літеру
4	Вивести слова введеного рядка довжиною більше 5-ти символів
5	Вивести слова введеного рядка у зворотному порядку (у словах порядок символів не змінювати)
6	Створити новий рядок зі слів введеного рядка, які містять дефіс
7	У введеному рядку замінити знаки арифметичних дій на їхні назви (‘+’ – на “plus” тощо)
8	У введеному рядку видалити усі символи, розміщені між першою та останньою крапками
9	У введеному рядку замінити знаки табуляції на пробіли та видалити зайві пробіли (початкові, кінцеві та подвійні)
10	У введеному рядку вставити своє власне ім’я після кожного знака оклику
11	У введеному рядку видалити усі слова, які містять цифри
12	У введеному рядку вставити після кожного слова його довжину
13	У введеному рядку видалити всі слова з довжиною менше 4-х символів
14	Обчислити кількість слів введеного рядка, які містять лише великі літери
15	Вивести всі слова введеного рядка, які містять лише маленькі літери
16	Обчислити кількість слів введеного рядка з парною довжиною
17	Вставити після кожного слова введеного рядка його порядковий номер
18	Вставити символ ‘*’ перед кожним словом введеного рядка
19	Вивести слова введеного рядка за абеткою
20	На основі введеного рядка створити новий рядок зі слів, які містять лише великі літери
21	Вивести слова введеного рядка, після яких у рядку йдуть розділові знаки
22	У введеному рядку поміняти місцями перше й останнє слово
23	У введеному рядку замінити розділові знаки на їх назви (кома, крапка тощо)
24	Визначити, скільки разів у введеному рядку зустрічається перше слово
25	З перших літер усіх слів введеного рядка створити новий рядок
26	Вивести слова введеного рядка разом з їхньою довжиною
27	У введеному рядку відсортувати слова за спаданням довжини
28	У введеному рядку видалити усі розділові знаки
29	Відсортувати слова введеного рядка за зростанням їх довжини
30	У введеному рядку поміняти місцями друге й останнє слова

Лабораторна робота № 20

Програмування з використанням структур (struct)

Мета роботи: набути практичних навиків організації структур і роботи з їхніми полями в Visual C++.

Теоретичні відомості

Поняття структури

Структура – це тип даних, який може поєднувати у собі різнотипні елементи. Елементи структури називаються *полями структури* і можуть мати будь-який тип, крім типу тієї самої структури, але можуть бути вказівником на нього.

Опис структури має синтаксис:

```
struct  <ім'я_типу_структури>
{ <тип1> <поле1>;
  <тип2> <поле2>;
   . . .
  <типN> <полеN>;
};
```

Наприклад, структуру `worker`, яка має чотири поля: прізвище співробітника (поле `name`), посада (поле `position`), вік (поле `age`) та зарплатня (поле `salary`), можна описати так:

```
struct worker
{ char name[15], position[10];
  int age; double salary;
};
```

Такий запис не виділяє пам'ять, а лише створює новий тип даних `worker`, ім'я якого може використовуватись при оголошуванні змінних. Наприклад, оголошення змінної `z`, масиву структур `Mas` та вказівника на структуру `ps`:

```
worker z, Mas[3], *ps;
```

При оголошуванні змінної типу структури виділяється пам'ять під усі поля структури послідовно для кожного поля. У наведеному прикладі структури `worker` під змінну `z` послідовно буде виділено 15, 10, 4, 8 байтів. Однак, розмір структури не завжди дорівнює сумі розмірів її полів, оскільки внаслідок вирівнювання об'єктів різної довжини у структурі можуть з'являтися безіменні "дірки". Правильне значення розміру допоможе визначити операція `sizeof`.

При оголошуванні структур їхні поля можна ініціалізовувати початковими значеннями, наприклад:

```
worker Mas[3] = {"Брончук","менеджер", 20, 2575.3,
                 "Ханін","програміст", 27, 34567,
                 "Яшкіна","бухгалтер", 50, 2699.99, };
```

Доступ до полів структури

Доступ до полів структури здійснюється за допомогою операцій вибору: операції "." (крапка) при звертанні до полів через ім'я структури і операції "->" при звертанні за вказівником, наприклад для вище оголошених змінних:

```
strcpy(z.name,"Бойко");
z.age = 34;
Mass[0].salary = 3750.5;
ps->salary = 4000;
```

До речі, для вказівника попередньо слід задати об'єкт посилання командою `ps = &z`, тобто записати адресу конкретної змінної типу структури.

Якщо елементом структури є інша структура, то доступ до її полів здійснюють через дві операції вибору:

```
struct A {int a; double x;};
struct B {A a; double x;} x[2];
x[0].a.a = 1;
x[1].x = 0.1;
```

Як видно з прикладу, поля різних структур можуть мати однакові імена, оскільки в них різна область видимості, головне при цьому не заплутатись.

Якщо полем структури є функція, то виклик цієї функції відбувається при звертанні до неї як до будь-якого іншого поля цієї структури. Наприклад, в структурі `student` одним з полів є функція `Show()`, яка будує рядок класу `String`, поєднуючи всі поля цієї структури для подальшого зручного виведення:

```
ref struct student
{ String ^prizv, ^gr;
  int math, inf;
  String^ Show()
   { return prizv+"  "+gr+"  "+math.ToString()+"  "+inf.ToString();
   }
};
```

В основній програмі цю функцію можна викликати в такий спосіб:

```
student z;
z.prizv = "Іванов";  z.gr = "ПІ-11";  z.math = 60;  z.inf = 90;
richTextBox1->Text = z.Show();
```

Дії над структурами

1) Для змінних одного й того самого структурного типу визначена *операція присвоювання*, при виконанні якої відбувається поелементне копіювання.

```
worker a, b = {"Хан","менеджер", 23, 1750};
a = b;
```

Команда `a = b` скопіює відразу всі чотири поля структури `b` в поля структури `a`.

```
struct  tovar
{ char  nazva[12],  virobnik[10];  int col; float vart; } ;
tovar x, y;
scanf("%s %s %i %g", x.nazva, x.virobnik, &x.col, &x.vart);
y = x;
printf("%s %s %i %5.2f\n", y.nazva, y.virobnik, y.col, y.vart);
```

2) Структуру можна передавати до функції та повертати в якості результату роботи функції.

```
Show(worker z)
{ cout<<z.name<<"  "<<z.position<<"  "<<z.age<<"  "<<z.salary<<endl;
}
worker Put()
{ worker w;
  cout<< "Введіть рядок з даними про робітника \n
         Прізвище  Посада Вік  Зарплатня" << endl;
  return cin >> w.name >> w.position >> w.age >> w.salary;
}
void main ()
{ date a, b = {"Лі","бухгалтер", 33, 2700};
  Show(b);   // виклик функції з передаванням структури b до функції
  a = Put(); // отримання структури в якості результату виконання функції
  Show(a);
  . . . . .
```

Приклади програм зі структурами

Приклад 1. Створити програму для опрацювання результатів сесії студентів: прізвище студента, група і результати двох екзаменів. У програмі передбачити можливість введення і виведення даних про довільну кількість студентів та відбір даних про студентів, які успішно здали сесію й вийшли на стипендію (не мають незадовільних оцінок, а середній бал склав понад 75 балів), також визначити прізвище студента з найбільшим середнім балом.

Розв'язок. У наведеному нижче програмному коді для одночасного введення всіх даних (чотирьох полів структури: прізвище, група і результати двох екзаменів) про одного студента використано функцію `scanf`, а виведення таких даних організовано за допомогою функції `printf`.

Ще однією особливістю цього програмного коду є почергове використання різних кодувань (команди `setlocale(0,".1251")` і `setlocale(0,".OCP")`) для коректного виведення літер кирилиці в консолі. Без використання цих команд кирилиця виводитиметься у вигляді абракадабри. Специфіка полягає у тому, що Visual C++ по-різному інтерпретує коди введених символьних даних і набраного у програмі тексту. Так, для виведення на екран текстових повідомлень слід використовувати кодування Windows 1251 (команда `setlocale(LC_ALL,".1251")` або `setlocale(0,".1251")`), а для виведення раніш введених символьних даних слід повернутися до початкових налаштувань командою `setlocale(0,".OCP")`.

Текст програмного коду консольного додатка:

```
#include <iostream>
using namespace std;
int main()
{
  setlocale(LC_ALL,".1251");
```

```
 struct student
 { char surname[22], gr[8];
   int ex1, ex2;
 } ;
int kol=0;           // загальна кількість студентів
cout<<"Введіть кількість  студентів - "; cin>> kol;
student *z = new student[kol];
cout<< "Введіть почергово рядки з відомостями про успішність " << kol
    << " студентів: \nПрізвище Група Оцінка1 Оцінка2" << endl;
for(int i=0; i<kol; i++)
  scanf("%s %s %i %i",z[i].surname,z[i].gr, &z[i].ex1,&z[i].ex2);
cout<< "\n   Успішність "<<kol
    << " студентів: \nПрізвище        Група Оцінка1 Оцінка2" << endl;
setlocale(0, ".OCP");
for(int i=0; i<kol; i++)
   printf("%s\t%s\t%i\t%i\n",z[i].surname,z[i].gr, z[i].ex1,z[i].ex2);
setlocale(0, ".1251");
cout<< endl << "Успішно здали сесію та вийшли на стипендію:" << endl;
int n=0;             // кількість стипендіатів
double sr,max(0);    // середній бал кожного та його максимальне значення
char maxname[22];    // прізвище студента з максимальним середнім балом
for(int i=0; i<kol; i++)
{ if(z[i].ex1>=60 && z[i].ex2>=60 && (z[i].ex1+z[i].ex2)/2.0 >= 75)
  { n++;
    setlocale(0, ".OCP");
    printf("%s\t%s\t%i\t%i\n",z[i].surname,z[i].gr,z[i].ex1,z[i].ex2);
  }
  sr=(z[i].ex1+z[i].ex2)/2.0;
  if(sr>max) { max=sr; strcpy(maxname,z[i].surname); }
}
setlocale(0, ".1251");
cout<< endl << "Кількість стипендіатів - " << n << endl;
cout<< endl << "Максимальний середній бал "<< max <<" має ";
setlocale(0, ".OCP"); cout<< maxname << endl << endl;
system ("pause>>void");
return 0;
}
```

```
Введіть кількість  студентів - 3
Введіть почергово рядки з відомостями про успішність 3 студентів:
Прізвище Група Оцінка1 Оцінка2
Костенко КТ-16  60      85
Халанчук ПІ-01  95      90
Мацкевич КТ-16  45      78

   Успішність 3 студентів:
Прізвище        Група Оцінка1 Оцінка2
Костенко        КТ-16   60      85
Халанчук        ПІ-01   95      90
Мацкевич        КТ-16   45      78

Успішно здали сесію та вийшли на стипендію:
Халанчук        ПІ-01   95      90

Кількість стипендіатів - 1

Максимальний середній бал 92.5 має Халанчук
```

Приклад 2. Створити програму для опрацювання даних телефонної книги (не менше п’яти абонентів): прізвище, ім’я, номер телефону, дата народження. Передбачити можливість виведення всіх даних телефонної книги та відбір даних про абонентів, номери яких розпочинаються з 067 або 068 (абоненти Київстар).

Розв’язок. У наведеному нижче програмному коді дані телефонної книги заповнюються самою програмою під час ініціалізації масиву структур.

Текст програмного коду консольного додатка:

```
#include <iostream>
#include <clocale>
using namespace std;
int main()
{ setlocale(0,".1251");
  struct abonent
  { char surname[21], name[15], tel[15], d[11];
  } z[]={"Харитонова","Іванна","066-99-55-444","11.05.1985",
         "Лавріненко","Петро","093-10-12-500","23.12.87",
         "Кононова","Олена","067-55-10-123","30.12.1990",
         "Петранчук","Олег","067-55-10-123","03.02.56",
         "Холоденко","Тетяна","068-333-11-850","15.08.79" };
 int kol = sizeof(z) / sizeof(abonent);
 cout<<"   Телефонна книга з "<<kol
     <<" абонентами: \nПрізвище  Ім’я  Номер телефона  Дата народження"<<endl;
 for(int i=0; i<kol; i++)
   cout<<z[i].surname<<"\t"<<z[i].name<<"\t"<<z[i].tel<<"\t"<<z[i].d<<endl;
 cout<<endl<<" Абоненти, номери яких розпочинаються з 067 або 068:"<<endl;
 int n=0;
 for (int i=0; i<kol; i++)
  if (!strncmp(z[i].tel,"067",3)||!strncmp(z[i].tel,"068",3))
   { n++;
     cout<<z[i].surname<<"\t"<<z[i].name<<"\t"<<z[i].tel<<"\t"<<z[i].d<<endl;
   }
 cout<<endl<<"Кількість абонентів Київстар - "<<n<<endl;
 system ("pause>>void");
 return 0;
}
```

Результати роботи консольного додатка:

```
   Телефонна книга з 5 абонентами:
Прізвище  Ім’я  Номер телефона  Дата народження
Харитонова      Іванна  066-99-55-444   11.05.1985
Лавріненко      Петро   093-10-12-500   23.12.87
Кононова        Олена   067-55-10-123   30.12.1990
Петранчук       Олег    067-55-10-123   03.02.56
Холоденко       Тетяна  068-333-11-850  15.08.79

 Абоненти, номери яких розпочинаються з 067 або 068:
Кононова        Олена   067-55-10-123   30.12.1990
Петранчук       Олег    067-55-10-123   03.02.56
Холоденко       Тетяна  068-333-11-850  15.08.79

Кількість абонентів Київстар - 3
```

Приклад 3. Ввести дані про товари: назва, фірма-виробник, кількість і ціна. Визначити сумарну вартість кожного з товарів фірми LG та середню вартість таких товарів.

Текст програмного коду консольного додатка:

```
#include "stdafx.h"
#include <clocale>
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
  struct  tovar
  { char  nazva[12], pr[10];  int cnt;  float vart; };
  Int  kol = 0;
  setlocale( LC_ALL, ".1251" );
  cout << "Введіть кількість товарів -  ";  cin >>  kol;
  tovar *z =  new  tovar[kol];
  cout << "Введіть почергово рядки з відомостями про " << kol
  << " товари(ів):\nНайменування Виробник Кіл-ть Ціна (десяткова кома)"<<endl;
  setlocale( LC_ALL, ".OCP" );
  for (int i=0; i<kol; i++)
    scanf("%s %s %i %g",z[i].nazva,z[i].pr,&z[i].cnt,&z[i].vart);
  double allsum=0, Svart, skol=0; int ktel=0;
  setlocale(LC_ALL, ".1251");
  cout << "\nПерегляд товарів фірми  LG:
          \n№№ Найменування Виробник  Кіл-ть  Ціна Вартість" << endl;
  setlocale(LC_ALL, ".OCP");
  for (int  i=0; i<kol; i++)
   if (!stricmp(z[i].pr,"LG"))
   { ktel++;
     skol + =z[i].cnt;
     Svart = z[i].vart * z[i].cnt;
     allsum += Svart;
     printf("%i  %s\t%s\t%i\t%5.2f\t%6.2f\n",
            ktel, z[i].nazva, z[i].pr, z[i].cnt, z[i].vart, Svart);
   }
  allsum /= skol;
  setlocale(LC_ALL, ".1251");
  printf("\nСередня вартість товарів фірми LG - %5.2f\n", allsum);
  delete []z;
  system ("pause>>void");
  return 0;
}
```

Результати роботи консольного додатка:

```
Введіть кількість товарів -  3
Введіть почергово рядки з відомостями про 3 товари(ів):
Найменування Виробник  Кіл-ть  Ціна (десяткова кома)
Телефон         LG       5       599,50
Телевізор      Sony      3      4199,99
Планшет         LG      10      2400,00

Перегляд товарів фірми  LG:
№№ Найменування Виробник  Кіл-ть  Ціна Вартість
1  Телефон      LG       5       599,50  2997,50
2  Планшет      LG       10      2400,00 24000,00

Середня вартість товарів фірми LG - 1799,83
```

Приклад 4. Ввести відомості про прізвища і дати народження людей та визначити наймолодшого з них.

Розв'язок. Налаштування форми рекомендуємо провести у такій послідовності.

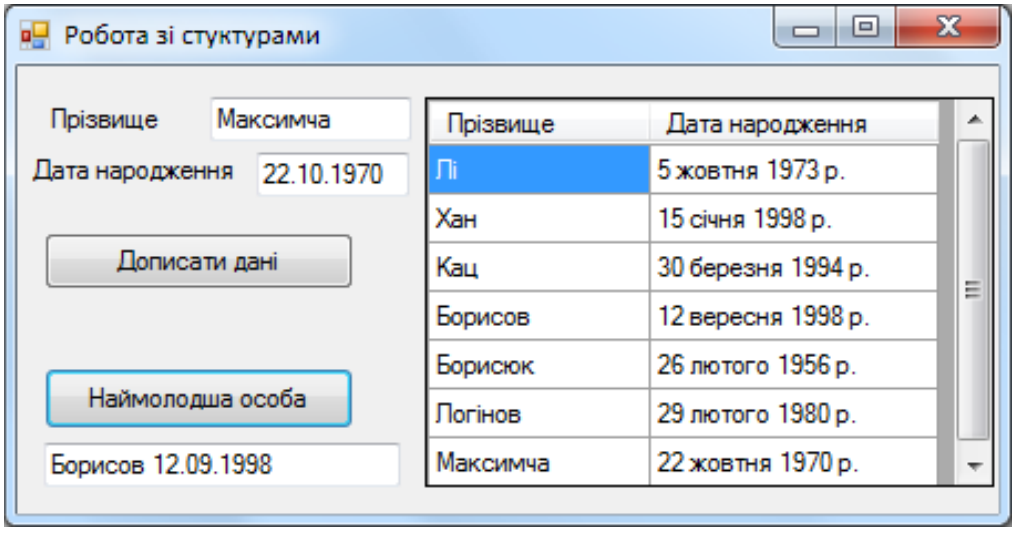
1) Розмістити на формі `dataGridView` і сформувати на ньому 2 стовпці (команда *Добавить столбец* у контекстному меню).

2) Командою контекстного меню *Правка столбцов* в однойменному діалоговому вікні для кожного стовпця задати текст заголовка (властивість *HeaderText*): *Прізвище* і *Дата народження*.

3) Задати для властивостей `AllowUserToAddRows` (дозвіл автоматичного додавання рядків) і `RowHeadersVisible` (відображення заголовка рядків) значення `False`.

4) Розмістити на формі два елементи `label`, два `textbox` і дві кнопки `button`. Задати надписи на них (властивість `Text`):

– для `Form1` – *Робота зі структурами*;
– для `label1` – *Прізвище*;
– для `label2` – *Дата народження*;
– для `button1` – *Дописати дані*;
– для `button2` – *Наймолодша особа.*



Текст програми:

```
ref struct person  // посилальний тип структури з двома полями
{ String^ prizv;
  DateTime  birth;
};
// глобальна змінна – кількість осіб, на початку має значення 0 і буде
// збільшуватись на 1 при натисканні кнопки “Дописати дані”
static int n=0;
// Кнопка “Дописати дані”
private:System::Void button1_Click(System::Object^ sender,System::EventArgs^e)
{
  person p;
  p.prizv = textBox1->Text;
  p.birth = DateTime::Parse(textBox2->Text);
  dataGridView1->Rows->Add();
  dataGridView1->Rows[0,n]->Value = p.prizv;
  dataGridView1->Rows[1,n]->Value = p.birth.ToLongDateString();
  n++;
}
// Кнопка “Наймолодша особа”
private:System::Void button2_Click(System::Object^ sender,System::EventArgs^e)
{
  int k=dataGridView1->Rows->GetRowCount(DataGridViewElementStates::Visible);
  person p;
  String^ young;  DateTime d, max=DateTime::Parse("01.01.1900");
  // або так max=DateTime::Parse(dataGridView1[1,0]->Value->ToString());
  for(int i=0; i<k; i++)
  { d = DateTime::Parse(dataGridView1[1,i]->Value->ToString());
   // або young = Convert::ToString(dataGridView1[0,i]->Value);
   if(max < d)
   { young = dataGridView1[0,i]->Value->ToString();
     max = d;
   }
  }
 textBox3->Text = young + " " + max.ToShortDateString();
}
```

Питання та завдання для самоконтролю

1) Дайте визначення структури як типу даних.

2) Як здійснюється доступ до полів структури?

3) Як визначається обсяг пам’яті, потрібний для зберігання структури?

4) Чи можуть в одній програмі збігатися імена полів структури і змінних?

5) Навести оголошення структури, яка описуватиме для деякого електричного приладу такі характеристики: назва приладу, споживана потужність і номінальна напруга.

Лабораторне завдання

1) У протоколі лабораторної роботи дати відповіді на контрольні питання.

2) У протоколі лабораторної роботи написати мовою С++ програмний код для формування 5 – 10 даних типу структури, поля якої подано в табл. 20.1 відповідно до індивідуального варіанта.

3) Створити на комп'ютері програмні проекти у середовищі Visual C++ для реалізації написаних програм.

Таблиця 20.1

Варіанти завдань для роботи зі структурами

№ вар.	Поля структури	Індивідуальні завдання
1	*Сесійні дані про студентів:* – прізвище, – група, – фізика – інформатика – історія	Визначити середній бал оцінок кожного студента і відібрати студентів, середній бал яких більше 75
2		Визначити середній бал оцінок усіх студентів з фізики і відібрати студентів, які склали екзамен з інформатики на "відмінно" (≥90)
3		Визначити кількість студентів, які не склали екзамен з інформатики, та визначити їх середній бал
4		Відібрати студентів-відмінників та визначити їхню кількість
5		Відібрати студентів, які здали сесію, але не "дотягують" до стипендії, та обчислити процентний вміст таких студентів серед решти представлених студентів
6		Відібрати студентів, які мають незадовільну оцінку хоча б з одного екзамену, та визначити їхню кількість
7	*Дані про працівників:* – прізвище, – посада, – освіта – рік народження, – зарплатня	Визначити працівників з найбільшою та найменшою зарплатнею і визначити їх сумарну зарплатню
8		Відібрати працівників, яким до пенсії (до 6-ти років) лишилося менше 3-х років, та обчислити їх процентний вміст серед решти працівників
9		Визначити наймолодшого і найстаршого працівників
10		Визначити працівників, які в поточному році святкують ювілей – вік кратний 5-ти чи 10-ти
11		Відібрати працівників молодших 30-ти років та обчислити їхню кількість
12		Відібрати працівників, зарплатня яких більше середнього значення зарплатні всіх працівників

Закінчення табл. 20.1

№ вар.	Поля структури	Індивідуальні завдання
13	*Товари на складі*: – найменування, – виробник, – ціна, – кількість	Визначити найдорожчий товар на складі, вивести всі дані про нього та обчислити його сумарну вартість
14		Обчислити загальну кількість та середню ціну товарів
15		Обчислити сумарну вартість кожного товару та сумарну вартість усіх товарів на складі
16		Відібрати товари, кількість яких менше 10-ти, обчислити кількість найменувань і сумарну кількість таких товарів
17		Визначити товар з найбільшою загальною вартістю на складі
18		Визначити товар з найбільшою кількістю на складі та обчислити його сумарну вартість
19	*Характеристики веб-сайту:* – повне доменне ім’я; – категорія (особистий, корпоративний, інформаційний, промо-сайт, блог, Інтернет-магазин, соціальна мережа тощо); – кількість відвідувань в день; – рейтинг PR (Page Rank – число від 0 до 10)	Відібрати сайти з рейтингом понад 5 та обчислити їхню кількість серед представлених
20		Відібрати сайти на доменах "ua" та "com", обчислити їхню кількість
21		Обчислити середній рейтинг та середню кількість відвідувань за день усіх представлених сайтів
22		Відібрати сайти з категорії "blog" та обчислити їхню кількість серед представлених
23		Відібрати сайти на домені "ua" з рейтингом PR понад 7 та обчислити їхню кількість серед представлених
24		Відібрати сайти з кількістю відвідувань понад 50 та обчислити кількість таких сайтів серед представлених
25		Визначити сайт з найбільшим рейтингом
26		Відібрати сайти на домені "net" або "ru", та обчислити їхню кількість
27	*Література у видавництві*: – автор; – назва книги; – рік видання; – тираж; – кількість сторінок	Відібрати книги з тиражем до 1000 екземплярів та обчислити кількість таких книг серед представлених
28		Відібрати книги, видані у поточному році та обчислити кількість таких книг
29		Визначити найтовстішу книгу (максимальна кількість сторінок)
30		Відібрати книги з кількістю сторінок понад 150 та обчислити кількість таких книг

Список рекомендованої літератури

1. С++. Теорія та практика: навч. посібник з грифом МОНУ/ [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : ВЦ ОНАЗ, 2011. – 587 с.

2. С++. Основи програмування. Теорія та практика: підручник / [О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред. О. Г. Трофименко. – Одеса : Фенікс, 2010. – 544 с.

3. Страуструп Б. Язык программирования С++. Специальное издание ; пер. с англ. / Страуструп Б. – М. : ООО "Бином-Пресс", 2006. – 1104 с.

4. Стивен Прата. Язык программирования С++. Лекции и упражнения : учебник : Пер с англ. / Стивен Прата. – СПб.: ООО "ДиаСофтЮП", 2005. – 1104 с.

5. Зиборов В. В. MS Visual C++ 2010 в среде .NET. Библиотека программиста / Зиборов В. В. – СПб. : Питер, 2012. – 320 с.

6. Хортон А. Visual C++ 2010: полный курс.; пер. с англ. / Хортон А. – М. : ООО "И.Д. Вильямс", 2011. – 1216 с.

7. Довбуш Г. Ф. Visual C++ на примерах / Г.Ф. Довбуш, А.Д. Хомоненко ; под ред. проф. А.Д. Хомоненко. – СПб. : БХВ-Петербург, 2007. – 528 с.

8. Visual C++ .NET: пособие для разработчиков C++ / [А. Корера, С. Фрейзер, С. Джентайл и др.] – М. : ЛОРИ, 2003. – 398 с.

9. Дейтел Х. М. Как программировать на С++; пер с англ. / Х. М. Дейтел, П. Дж. Дейтел – М. : ООО "Бином-Пресс", 2008. – 1456 с.

10. Трофименко О. Г. Алгоритмізація обчислювальних процесів і особливості програмування в С++ : метод. посібник. – Модуль 4. – / Трофименко О. Г., Буката Л. М., Леонов Ю. Г. – Ч. 2. – Одеса : ІЦ ОНАЗ, 2009. – 93 с.

ЗМІСТ