

ТЕМА: Методи роботи з функціями.

Вбудовані функції

Вбудована функція – це така, відтрансльований код якої компілятор підставляє у місці виклику замість підстановки команд виклику. Вбудовані функції виконуються трохи швидше від звичайних, однак потребують додаткової пам'яті.

Вбудовані функції зазвичай є короткими, простими, але функціонально виокремленими. Наприклад: обчислити середнє геометричне значення трьох дійсних чисел; з трьох різних чисел обрати середнє. Щоб замінити виклик функції підстановкою, найпершим словом означення функції має бути `inline`:

```
// визначення вбудованих функцій
inline int pmsign(double value) // знак числа
{
    return value >= 0 ? +1 : -1 ;
}
inline double meangeo3(double a, double b, double c)
{ // середнє геометричне
    return pow(abs(a*b*c), 1.0/3.0)*pmsign(a)*pmsign(b)*pmsign(c);
}
inline int medium(int x, int y, int z) // серединне
{
    return x < y && x > z ? x : (y < x && y > z ? y : z) ;
}
```



```
// використання вбудованих функцій
int main()
{
    double dt = meangeo3(1.5, 2.0, -10.0);
    // компілятор виконає підстановки:
    // double dt = pow(abs(1.5*2.0*(-10.0)), 1.0/3.0) *
    // (1.5>=0 ? +1 : -1)*(2.0>=0 ? +1 : -1)*(-10.0>=0 ? +1 : -1) ;
    cout << dt << endl;
    int plus = medium(2,8,3) + medium(1,6,2);
    // підстановки аналогічні
    cout << plus << endl;
    system("pause");
    return 0;
}
```

Результати:

```
-3.10723
5
```

З точки зору компілятора програма виглядає так, як показані в **тексті підстановки**.

Зауважимо, що компілятор не зобов'язаний надати функції статусу вбудованої, наприклад, у випадку надмірного розміру коду або рекурсії.

Аргументи за замовчуванням

Це є значення, які автоматично використовують у випадку відсутності аргумента при виклику функції. Значення за замовчуванням записують дуже подібно до ініціалізації змінної величини. Наприклад, якщо оголошення функції визначає один аргумент за замовчуванням:

```
void anyfunc(double x=0.0)
{
    // . . . . .
}
```

тоді виклик цієї функції можна виконати двома способами:

```
anyfunc(5.62); // явне значення аргумента x
anyfunc();     // функція використовує значення за замовчуванням
```

Механізм аргументів за замовчуванням дозволяє спростити використання функцій з великою кількістю параметрів. Частина параметрів треба добре продумати і надати їм значень за замовчуванням.

Приклад. Скласти функцію, яка в заданому числовому масиві переставляє на початок елементи або від'ємні, або нульові, або додатні, зсуваючи решту праворуч.

Отже, функція повинна мати три аргументи: вказівник на масив, розмір заданого масиву, знак переставлених елементів. Перші два аргументи завжди будуть різними, а третій аргумент буде повторюватись: -1 для від'ємних, 0 для нульових, +1 для додатних. Можна припустити, що частіше треба переставляти на початок від'ємні елементи, отже третьому аргументу можна надати значення -1 за замовчуванням.

```

void selection(int * m, int size, int sign=-1); // прототип

int main()
{
    const int num=12; // розмір масиву
    int data[num] = { 4, 0, -5, 0, 2, 1, -9, -3, 0, -7, 16, 8 };
    selection(data,num); // третій аргумент - за замовчуванням
    for(int i=0; i<num; i++) cout << data[i] << " "; // результат
    cout << endl;
    // selection(data,num,0); // приклад: явний третій аргумент
    // третій аргумент - нульові елементи
    // for(int i=0; i<num; i++) cout << data[i] << " "; // результат
    cout << endl;
    system("pause");
    return 0;
}

```


Результат для третього аргумента за замовчуванням:

```
-5  -9  -3  -7  4  0  0  2  1  0  16  8
```

Додаткові правила. 1) Параметри, які отримують значення за замовчуванням, мають бути в списку всіх параметрів **правіше звичайних аргументів** – тобто, від кінця списку. 2) Значення за замовчування записують лише в прототипі, а в заголовку – ні, якщо прототип визначений окремо. Якщо ж прототипу немає, тоді записують в заголовку як звичайно.


```
// повне визначення функції
void selection(int * m, int size, int sign)
// в заголовку ще раз за замовчуванням не треба
{
    int * temp = new int[size]; // будуємо тимчасовий допоміж. масив
    int k=0; // лічильник переставлених елементів
    int i;
    for(i=0; i<size; i++) // перший перегляд - переставити на початок
        switch ( sign )
        {
            case -1: if (m[i]<0) temp[k++] = m[i]; break;
            case 0:  if (m[i]==0) temp[k++] = m[i]; break;
            case +1: if (m[i]>0) temp[k++] = m[i]; break;
        }
    for(i=0; i<size; i++) // другий перегляд - зсув решти
        switch ( sign )
        {
            case -1: if (m[i]>=0) temp[k++] = m[i]; break;
            case 0:  if (m[i]!=0) temp[k++] = m[i]; break;
            case +1: if (m[i]<=0) temp[k++] = m[i]; break;
        }
    // копіюємо назад в масив
    for(i=0; i<size; i++) m[i]=temp[i];
    delete [] temp; // звільнити допоміжний масив
}
```

Перевантаження функцій

 Перевантаження – це використання одного імені для декількох функцій. Зазвичай перевантажені функції виконують одну й ту ж саму роботу, але зобов'язані мати різні списки аргументів (сигнатуру), тобто, або інші типи аргументів, або іншу кількість. В цьому разі компілятор може визначити, який варіант функції потрібно викликати.

Зауваження. Статус `const` або інший статус включається до сигнатури, тип результату функції – не включається.

Приклад. Для довільного масиву є потреба його ініціалізувати трьома способами: всі елементи мають фіксоване задане значення; читати значення елементів з файла; читати значення елементів з клавіатури.

В подібних випадках є сенс перевантажити функцію ініціалізації трьома способами відповідно до задачі. У всіх випадках перші два параметри визначають вказівник на масив і його розмір. Третій параметр: значення для ініціалізації – число; ім'я файла – символьний рядок; для читання з клавіатури – відсутній. Отже, сигнатури всіх трьох перевантажених функцій будуть різними.

```
#include <iostream>
#include <fstream>
using namespace std;
// ініціалізація масиву різними способами
// прототипи перевантажених функцій
void msinit(float * x, int size, float value);
// фіксоване значення
void msinit(float * x, int size, char * filename);
// читати з файла
void msinit(float * x, int size); // читати з клавіатури

int main()
{
    const int num=12; // розмір масиву
    float data[num]; // пам'ять для масиву
    msinit(data,num,9.23);
    //msinit(data,num,"test1.txt");
    //msinit(data,num);
    for(int i=0; i<num; i++)    cout << data[i] << "    ";
    cout << endl;
    system("pause");
    return 0;
}
```



```
// визначення перевантажених функцій

void msinit(float * x, int size, float value)
    // фіксоване значення
{
    for(int k=0; k<size; k++) x[k]=value;
}

void msinit(float * x, int size, char * filename)
    // читати з файла
{
    ifstream inp(filename);
    for(int k=0; k<size; k++)    inp >> x[k];
    inp.close();
}

void msinit(float * x, int size)    // читати з клавіатури
{
    cout << "print " << size << " float numbers:\n";
    for(int k=0; k<size; k++)    cin >> x[k];
}
```

Висновок. Перевантажені функції зазвичай виконують однакову роботу, але за різними алгоритмами.

Випадки неоднозначності перевантажених функцій

Можна створити ситуацію, в якій компілятор не зможе обрати одну з перевантажених функцій. Такі випадки називають неоднозначними і породжують помилку компіляції.

Неоднозначні ситуації найчастіше можуть виникати через аргументи перевантажених функцій, задані за замовчуванням.

Якщо, наприклад, в попередньому прикладі визначити заголовок для випадку фіксованого значення так:

```
void msinit(float * x, int size, float value=0.0); // фікс.знач.
```

тоді виклик

```
msinit(data, num);
```

не дозволяє обрати між

```
void msinit(float *,int) // читати з клавіатури ?
```

```
void msinit(float *,int,float) // фіксоване значення ?
```

Другою причиною неоднозначності перевантажених функцій може бути автоматичне перетворення типів. Якщо, наприклад, є дві перевантажені функції, одна з аргументом типу `float`, а друга з аргументом типу `double`, тоді виклик функції з фактичним аргументом `int` породжує неоднозначність: в який тип перетворити `int`?

Третя причина неоднозначності – коли одна функція має аргумент `x`, а друга – неявний вказівник (посилання) `&x`. При виклику функції не буде синтаксичної різниці щодо вибору функції.

Шаблони функцій

За допомогою шаблону можна створити узагальнену функцію, яка працює з типом даних, заданим як параметр. Ту саму функцію можна застосувати до різних типів даних, не будуючи окремі перевантажені варіанти для кожного типу.

Шаблонна функція має починатись заголовком `template` і має один або більше параметрично заданих типів даних.

Приклад. Скласти шаблонну функцію, яка обчислює кількість елементів заданого інтервалу, які входять в числовий масив або символьний рядок.

```
// визначення шаблонної функції
// заголовок
template <typename TN> int goin(TN * obj, int size, TN a, TN b)
{ // TN - параметричний тип даних
  int count=0;
  for(int i=0; i<size; i++) if(obj[i]>=a && obj[i]<=b) count++;
  return count;
}
```


генерування – за
типами параметрів

```
int main()
{
    int avec[10] = { 4,5,6,-5,-2,6,9,9,1,4 };
    double bvec[8] = { 0.1, -2.2, 6.0, 4.3, -5.9, -6.4, 8.7, 3.0 };
    char cvec[] = "abcbdfgklcfdmqrui";
    int k1 = goin(avec, 10, -5, 5); // TN = int
    int k2 = goin(bvec, 8, 0.0, 7.0); // TN = double
    int k3 = goin(cvec, strlen(cvec), 'c', 'g'); // TN = char
    cout << k1 << '\t' << k2 << '\t' << k3 << endl; // 6 4 7
    system("pause");
    return 0;
}
```

Сам по собі шаблон не створює ніяких функцій. Для шаблонної функції компілятор автоматично генерує стільки різних варіантів, скільки є способів її виклику. Шаблон функції фактично означає, що функція перевантажує сама себе.

Проте потрібно бути уважним, бо операції шаблонної функції можуть бути незастосовними до певних типів даних, що породить помилку компіляції згенерованої функції.


Шаблонні функції також можна перевантажити, якщо скласти іншу сигнатуру.



Задачі і алгоритми

Задача 1. Прямокутна таблиця має розмір M рядків і N стовпців. В кожній клітці таблиці записана деяка буква малою літерою або літера «пропуск». Порахувати, скільки разів в цій таблиці зустрічається слово «задача». Словом вважати послідовність букв в сусідніх клітках, розташованих горизонтально зліва направо, вертикально зверху вниз, діагонально зліва вгору направо або зліва вниз направо. Наприклад, для випадку букв з числа лише вказаних слів, може бути так:

		а						а
	з	з	ч				ч	
		а		а		а		
		д	д		д			
	з	а	д	а	ч	а		
		ч	з		ч		з	
з	з	а	д	а	ч	а		ч

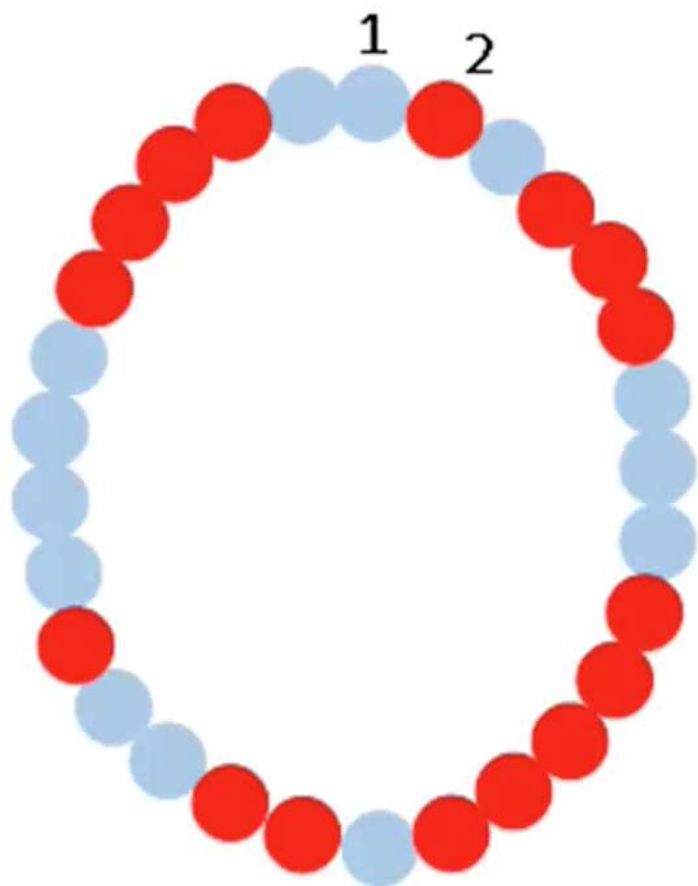


Відповідь: 5

Задано: M , N , матриця $M \times N$ літер в текстовому файлі.

Знайти: кількість слів «задача».

Задача 2. Маємо буси, що складаються з N бусинок ($N \leq 100$). Бусинки можуть бути червоного або голубого кольору. Наприклад, для $N=29$:



Цифрами позначені позиції першої і другої бусинок.

Конфігурація бусинок задана послідовністю кольорів бусинок ("b" – голубий, "r" – червоний), починаючи з бусинки номер 1. Наприклад, зображені на рисунку буси задані такою послідовністю букв:

b r b r r r b b b r r r r r b r r b b r b b b b r r r r b

Необхідно розірвати буси і потім знімати бусинки одного кольору з одного кінця, поки не зустрінеється бусинка іншого кольору. Те ж саме зробити з другим кінцем (бусинки, зняті з різних кінців, можуть бути різного кольору).

Визначити точку такого розриву заданих бусів, щоб сумарна кількість бусинок, зібраних з обидвох кінців, була максимальна.

Наприклад, для бусів на малюнку, точка розриву може бути між 24 і 25 бусинками або між 9 і 10 бусинками, при цьому сумарна кількість бусинок в обидвох випадках дорівнює 8.

Задано: в текстовому файлі конфігурація бусів як послідовність букв без пропусків.

Знайти: M – максимальне число зібраних бусинок і розташування однієї з оптимальних точок розриву.

Задача 3. Схема автобусних сполучень між містами має форму матриці, заповнену назвами міст, наприклад (фрагмент):

Львів	Стрий	Тернопіль	...	Н
Київ	Черкаси	Н	...	Полтава
Рівне	Київ	Житомир	...	Н
...
Стрий	Дрогобич	Львів	...	Миколаїв

Розмір матриці фіксований 10×10 .

Перший стовпець – початковий пункт відправлення з вказаного міста.

В кожному рядку в позиціях з другої по десяту – назви міст, до яких є прямий автобусний маршрут. «Н» - позиція не заповнена.

Задано:

- 1) в текстовому файлі 10 рядків, в кожному рядку файла через кому список міст як в рядку таблиці;
- 2) назва міста М, від якого починається подорож.

Знайти: перелік міст, до яких можна доїхати прямо або з пересадками від заданого міста М.