Лабораторне заняття №8 (2-й семестр, 2022)

Шаблони функцій та класів в С++.

Шаблони (template) - засіб мови С ++, призначений для кодування узагальнених алгоритмів, без прив'язки до деяких параметрів (наприклад, типів даних).

1. Шаблони функцій

Шаблони функцій (шаблонна функція, узагальнена функція) - це узагальнений опис поведінки функцій, яка може бути викликана для об'єктів різних типів. Шаблон функції являє собою сімейство різних функцій (або опис алгоритму).

За описом шаблон функції схожий на звичайну функцію: різниця в тому, що деякі елементи не визначені (типи, константи) і є параметризовані.

template <typename або class параметр> //може бути декілька параметрів

заголовок функції

{тіло функції}

Шаблони функцій не компілюються безпосередньо. Коли компілятор зустрічає виклик шаблону функції, він копіює шаблон функції і замінює типи параметрів шаблону функції фактичними (переданими) типами даних.

Функція з фактичними типами даних називається екземпляром шаблону функції (або «об'єктом шаблону функції»).

Шаблони функцій працюють як з фундаментальними типами даних (char, int, double і т.д.), так і з класами (але повинні бути визначені операції для об’єктів класу, які використовуються в шаблоні).

Шаблони функцій також можуть бути перевантажені. Тоді вони повинні відрізнятись списками параметрів.

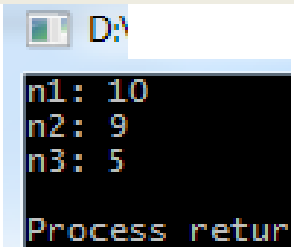
Шаблони функцій можуть мати також параметри за замовчуванням:

template<class T1=double, class T2=int>

Приклад 1. Шаблон функції додавання.

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

//шаблон функції (трафарет)  для додавання чисел різного типу
 template<typename T> //або <class T>, задавання параметру
  T add(T x, T y) //заголовок функції
   {
    return x + y;
   }

int main()
{
   double a1 = 4.7, b1 = 5.3;
   double n1 = add(a1, b1); //заміна параметра на тип int у функції add()

   int a2 = 4, b2 = 5;
   int n2 = add(a2, b2);//заміна параметра  на тип int у функції add()
```

```
   short int a3 = 3, b3 = 2;
   short int n3 = add(a3, b3); //заміна параметра на тип short int у функції add()

   cout << "n1: " << n1 << endl;
   cout << "n2: " << n2 << endl;
   cout << "n3: " << n3 << endl;

    return 0;
}
```



2. Шаблони класів

Шаблони класів – узагальнений опис класу (типу даних, який створює користувач), в якому можуть бути параметризовані атрибути і операції типу. Являють собою конструкції, за якими можуть бути згенеровані дійсні класи шляхом підстановки замість параметрів конкретних аргументів.

Створення шаблону класу аналогічно створенню шаблону функції:

template <class T> //або список параметрів

//також можуть бути параметри за замовчуванням

class Ім'я

{ ... };

Інстанціювання шаблона класа – створення конкретних класів з шаблону.

Приклад 2. Шаблон класу.

```
#include <iostream>
using namespace std;
template <class T>
class Point
{private:
    T x,y;
public:
    Point(T x, T y)
     {
         this->x=x; this->y=y;
     }
     void show();
};
//при багатофайловому проекті  визначення методів шаблону краще розміщувати
```

```
//в заголовковому файлі
template <class T> //визначення методів поза шаблона класу
void Point<T>::show()
 {
   cout<<"type x: "<<typeid(x).name()<<endl;
   cout<<"type y: "<<typeid(y).name()<<endl;
   cout<<"x= "<<x<< " y= "<<y<<endl;
 }
int main()
{//створюючі об'єкти класу, визначаємо тип параметра шаблону
    Point<int> a(1,1);
    a.show();
    Point<double> b(2.2,1.1);
    b.show();
    Point<char> c('a','b');
    c.show();
    return 0;
}
```

3. ЗАВДАННЯ на лабораторну роботу.

Змінити розроблену програму (згідно свого варіанту) для Лаб.раб.7, замінюючи клас Vector на шаблон класу (для різних типів даних). Залишити клас error.

Реалізувати роботу з об'єктами різних типів даних і обробку виключних ситуацій для них. (див.Лаб.раб.7, лекція 6,7).

ДОДАТКОВІ завдання (за бажанням, на додаткові бали):

1.Додати часткову або повну спеціалізацію шаблону класу (див.лекцію 7) ;

2.Додати ієрархію класів виключень з застосування віртуальних методів (див.лекцію 6).