ЛАБОРАТОРНАЯ РАБОТА № 3.
Шаблоны классов

Цель работы: изучить приемы создания и использования шаблонов классов.

Краткие теоретические сведения

Механизм **шаблонов** C++ – это средство построения обобщенных определений функций и классов, которые не зависят от используемых типов данных. Этот механизм позволяет сократить трудоемкость создания программ, т.к. повышает лаконичность текста, т.е. использование шаблонов избавляет от необходимости дублировать коды классов и функций для разных типов данных.

Компилятор по заданному типу аргументов на основе описания шаблона автоматически создает соответствующие экземпляры классов и функций, которые называются ***представители*** конкретных классов и функций.

Шаблоны класса в отличие от шаблона функции позволяют параметризовать, т.е. использовать в качестве параметров не только типы элементов данных, но и константы разных типов.

Синтаксис определения шаблона класса следующий:

template <список параметров шаблона > **обычное описание структуры**

При этом список параметров шаблона не может быть пустым. Элементы в списке разделяются запятыми. Внутри пространства класса параметры шаблона должны быть хотя бы один раз упомянуты.

В список параметров могут входить два вида параметров:

1) типизированные параметры, начинающиеся со слов

class Идентификатор ,

компилятор при создании экземпляра класса заменит его на конкретный тип данных;

2) нетипированные параметры шаблона:

Стандартный тип числовых данных ***Идентификатор***

Таким образом, типированные параметры – это фиктивные имена типов данных, входящих в класс. Нетипированные параметры – это поименованные типы числовых констант. При этом им при декларировании можно присваивать умалчиваемые значения.

Каждый параметр является локальным в рамках пространства класса.

В описании класса хотя бы один раз необходимо упомянуть ID типированных и нетипированных параметров, конкретные значения для которых будут переданы в момент создания объекта этого класса через список аргументов, который указывается через запятые в треугольных скобках сразу после ID класса:

ID_ класса <список аргументов> ID_объектов;

ID_объектов – идентификаторы объектов, которые создает данный класс (записанные через запятые, например *a,b,c*). При этом в списке аргументов каждому типированному параметру шаблона должен соответствовать известный конкретный тип данных, а каждому нетипированному параметру – константное выражение соответствующего типа. Таким образом, между списком параметров шаблона и списком аргументов должно быть абсолютное соответствие по количеству, порядку их следования и типам. Если нетипированные параметры имеют умалчиваемые значения, их располагают в списке последними.

Пример с шаблоном класса, конструктор которого порождает объекты с двумя параметризованными полями:

```
        . . .
template <class T1>
        class X {
                protected:
                T1 a, b;
                public:
                X(T1 i, T1 j) {                 // Конструктор
                        a = i; b = j;
                        cout << "\n Object created, size = " << sizeof(T1) << endl;
                        }
                void Print(void) {
                        cout << " a = " << a << "  b = " << b << endl;
                        }
                };
void main(void) {
        X < int >  x1(2, 3);                    // Целочисленные объекты
        x1.Print();                             // На экране: Object created, size = 2
        getch();                                //                a = 2  b = 3
        X < double >  x2(0.5, 1.2);             // Вещественные объекты
        x2.Print();                             // На экране: Object created, size = 4
        getch();                                //                a = 0.5  b = 1.2
        }
```

Методы шаблона класса можно определять ***вне его пространства***. В этом случае формат их определения будет следующим:

```
template <список параметров шаблона> тип результата ID класса <перечень через «,» ID из списка параметров шаблона> :: ID метода список параметров метода)
                {
                        код метода
                }
```

В вышеприведенном примере вынесем за пределы пространства класса код конструктора:

```
                . . .
        template <class T1>
                class X {
                        protected:
                        T1 a, b;
                        public:
                        X(T1, T1);
                        void Print(void) {
                                cout << " a = " << a << "  b = " << b << endl;
                                }
                        };
// Определение конструктора вне состава класса
        template <class T1> X <T1> :: X(T1 i, T1 j)
                {a = i; b = j;
                cout << " Object created, size = " << sizeof(T1) << endl;
                        }
        void main(void) {
                        // Код предыдущего примера
                }
```

Задание к лабораторной работе № 3

Общая постановка. Даны: число N и последовательность a_1, a_2, … a_N .

Создать шаблон класса, порождающего динамические одномерные массивы с элементами различных типов (вещественные, целочисленные, символьные и т.д.). Тип данных и результат являются параметрами по отношению к классу. Программа должна содержать: конструктор, деструктор, метод просмотра значений созданного массива, а также метод для решения задач формирования нового массива по соответствующему индивидуальному заданию.

Индивидуальные задания

1. a_1, $(a_1 + a_2)$, … , $(a_1 + a_2 + \ldots + a_N)$.
2. $(a_1 * a_1)$, $(a_1 * a_2)$, … , $(a_1 * a_N)$.
3. $|a_1|$, $|a_1 + a_2|$, … , $|a_1 + a_2 + \ldots + a_N|$.
4. a_1, $-a_1 * a_2$, $+a_1 * a_2 * a_3$, … , $(-1)^{N-1} * a_1 * a_2 * \ldots a_N$.
5. $-a_1$, $+a_2$, $-a_3$, … , $(-1)^N * a_N$.
6. $(a_1 + 1)$, $(a_2 + 2)$, $(a_3 + 3)$, … , $(a_N + N)$.
7. $a_1 * 1$, $a_2 * 2$, $a_3 * 3$, ... , $a_N * N$.
8. $a_1 * a_2$, $a_2 * a_3$, … , $a_{N-1} * a_N$. 9. $a_1/1$, $a_2/2$, $a_3/3$, … , a_N /N .
10. $(a_1 + a_2)$, $(a_2 + a_3)$, … , $(a_{N-1} + a_N)$.
11. $(a_1 + a_2 + a_3)$, $(a_2 + a_3 + a_4)$, $(a_3 + a_4 + a_5)$, …, $(a_{N-2} + a_{N-1} + a_N)$.
12. $(N + a_1)$, $(N - 1 + a_2)$, ... , $(1 + a_N)$.
13. $N * a_1$, $(N-1) * a_2$, ... , $(1 * a_N)$.
14. a_1/N, a_2/N, … , $a_N /1$.
15. $-a_1$, $+a_1 * a_2$, $-a_1 * a_2 * a_3$, … , $(-1)^N * a_1 * a_2 * \ldots a_N$.