

Министерство высшего образования и науки РФ
ФГБОУ ВО ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Информационная безопасность систем и технологий»

ОТЧЁТ
о лабораторной работе №6
Шаблоны в C++

Дисциплина: Языки программирования

Группа: 18ПИ1

Выполнил: Асаян А.В.

Количество баллов:

Дата сдачи:

Проверил: к.т.н., доцент Лупанов М.Ю.

Пенза, 2019

1 Цель работы

1.1 Освоить создание и использование шаблонов функций и шаблонов классов в программах на языке Си++.

2 Задания к практической работе

2.1 Реализовать алгоритм сортировки (алгоритм выбрать самостоятельно) в виде шаблонной функции. Продемонстрировать работу шаблонной функции на массивах различных типов.

2.2 Реализовать класс `Rectangle` с атрибутами, хранящими высоту и ширину, и поддерживающий сравнение по площади. Продемонстрировать сортировку массива объектов типа `Rectangle` шаблонной функцией, разработанной в предыдущем задании.

2.3 Реализовать шаблонный класс `DoubleBox` для хранения двух атрибутов разного типа. Тип атрибутов задать параметрами шаблона. Реализовать в классе конструктор по умолчанию, инициализирующий конструктор, а также методы `get` и `set`.

2.4 Задание повышенной сложности. Реализовать шаблонный класс `Array` для хранения массива

произвольного типа и размера (без использования динамической памяти).

Тип элементов и размер

массива задать параметрами шаблона. Реализовать в классе следующие конструкторы:

- конструктор по-умолчанию, без параметров;
- конструктор, позволяющий инициализировать весь внутренний массив одинаковыми значениями, имеющий один параметр — инициализирующее значение;
- конструктор, позволяющий инициализировать внутренний массив значениями из внешнего массива, имеющий два параметра — указатель на внешний массив и размер внешнего массива.

Реализовать в классе Array перегрузку оператора индексации operator[], для того чтобы можно было применять его к экземплярам класса для выполнения обращения к элементам внутреннего массива. Проверить работоспособность оператора индексации для константных объектов, а также работоспособность при использовании слева от оператора присваивания.

3 Результат выполнения работы

3.1 Для реализации в виде шаблонной функции был выбран алгоритм сортировки «сортировка пузырьком». В качестве параметров функции передаётся указатель на массив и количество элементов в массиве. В алгоритме происходят повторяющиеся проходы по массиву, элементы массива сравниваются с друг другом попарно, если левый элемент больше, то они меняются местами. Код программы:

```
#include <iostream>
using namespace std;
template <typename T> void sort (T* arr, int len){
    T tmp;
    for(int i = 0; i<len; i++){
        for(int j = (len-1); j>=(i+1); j--){
            if(arr[j]<arr[j-1]){
                tmp = arr[j];
                arr[j]=arr[j-1];
                arr[j-1]=tmp;
            }
        }
    }
}
int main(int argc, char **argv)
{
    int I[5]{5,4,6,9,11};
    double D[5]{3.2,3.4,4.5,0.0,99.1};
    char C[5]{'b','v','e','k','c'};
    sort(I,5);
    sort(D,5);
    sort(C,5);
    for(int i=0; i<5; i++){
        cout<<I[i]<<" ";
    }
}
```

```

        cout<<endl;
    for(int i=0;i<5;i++){
        cout<<D[i]<<" ";
    }
        cout<<endl;
    for(int i=0;i<5;i++){
        cout<<C[i]<<" ";
    }
        cout<<endl;
    return 0;
}

```

Результаты работы программы для трёх массивов типов int, double и char представлены на рисунке 1.

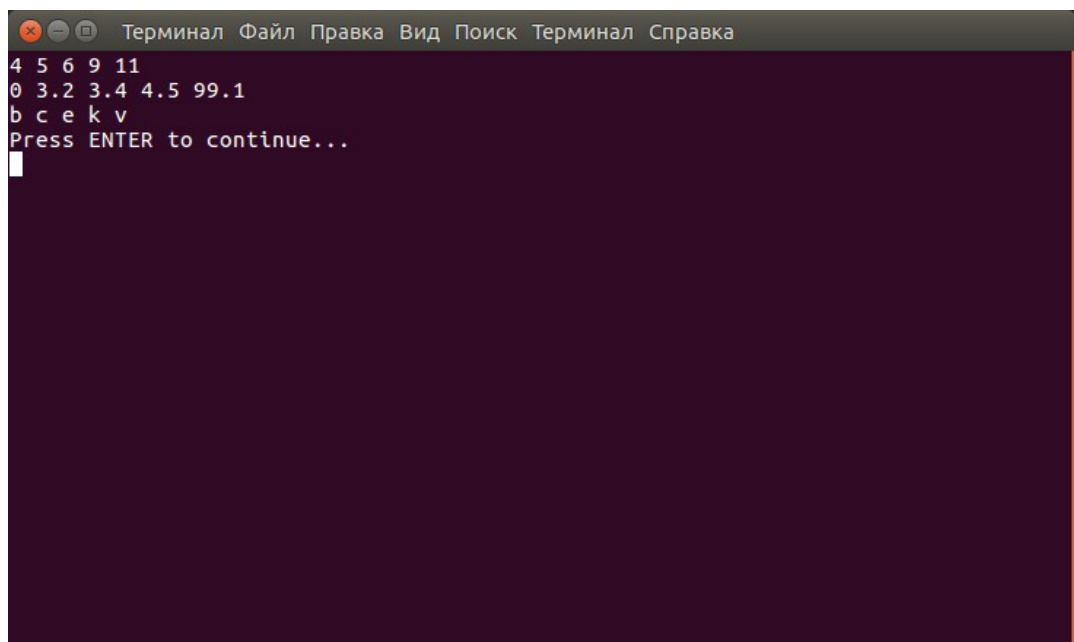


Рисунок 1 — Результаты работы сортировки.

3.2 Был реализован класс Rectangle с атрибутами хранящими высоту и ширину. Для него были перегружены операторы > и <, а также оператор присваивания = и оператор <<. Код программы:

```

#include <iostream>
using namespace std;
template <typename T> void sort (T* arr, int len){
    T tmp;
    for(int i = 0;i<len;i++){
        for(int j = (len-1);j>=(i+1);j--){
            if(arr[j]<arr[j-1]){
                tmp = arr[j];

```

```

        arr[j]=arr[j-1];
        arr[j-1]=tmp;
    }
}
}
class Rectangle
{
public:
    double width;
    double height;
    Rectangle(){
        width=0;
        height=0;
    };
    Rectangle(double w,double h) {
        width=w;
        height=h;
    };
    friend bool operator >(Rectangle a,Rectangle b)
    {
        if(a.width*a.height>b.width*b.height)
            return true;
        else
            return false;
    }
    friend bool operator <(Rectangle a,Rectangle b)
    {
        if(a.width*a.height<b.width*b.height)
            return true;
        else
            return false;
    }
    Rectangle &operator =(Rectangle & b)
    {
        width=b.width;
        height=b.height;
        return *this;
    }
    friend ostream& operator<< (ostream &out, const
Rectangle &b){
        out << "WIDTH " << b.width << ", HEIGHT " <<
b.height;
    }
}

```

```
};
int main(int argc, char **argv)
{
    Rectangle a(20,20);
    Rectangle b(4,5);
    Rectangle c(9,2);
    Rectangle p[3]{a,b,c};
    sort(p,3);
    for(int i=0;i<3;i++)
        cout<<p[i]<<endl;
    return 0;
}
```

Результаты работы программы для массива из трёх экземпляров класса Rectangle представлены на рисунке 2.

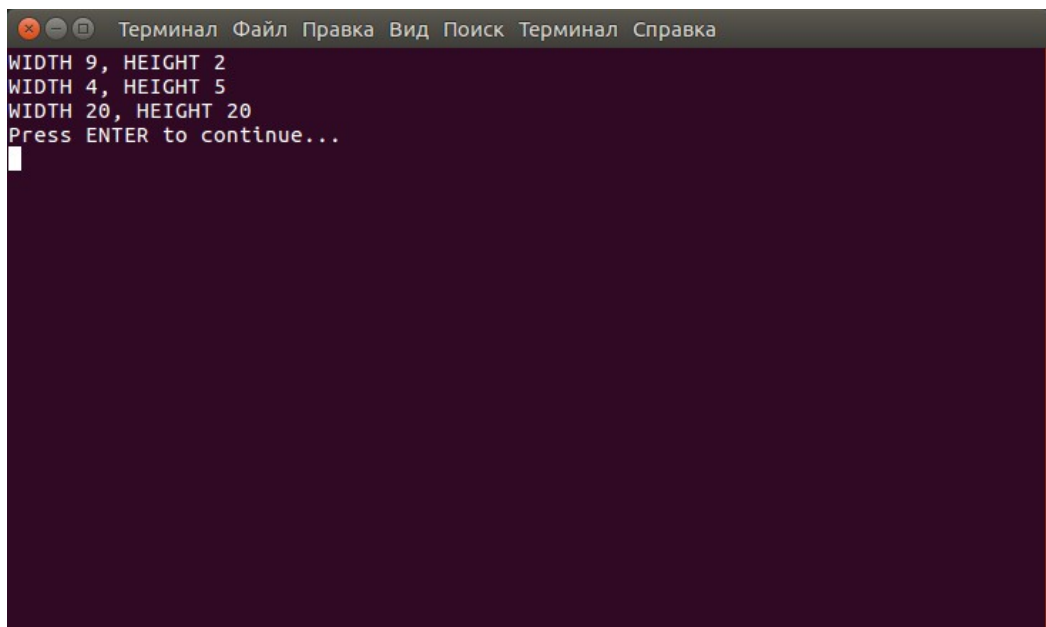


Рисунок 2 — Результат сортировки прямоугольников по площади

3.3 Был реализован шаблонный класс для хранения двух атрибутов разного типа DoubleBox. Код программы:

```
#include <iostream>
#include <string>
using namespace std;
template <typename T1, typename T2> class DoubleBox
{
private:
    T1 a = T1();
    T2 b = T2();
public:
```

```

        DoubleBox() {}
        DoubleBox(const T1 value1, const T2
value2):a(value1),b(value2) {}
        void getContent(const T1 value1,const T2 value2);
        void setContent(const T1 value1, const T2
value2);
    };
    template <typename T1, typename T2> void
DoubleBox<T1,T2>::getContent(T1 value1,T2 value2)
    {
        value1=a;
        value2=b;
        cout<<value1<<" "<<value2<<endl;
    }
    template <typename T1, typename T2> void
DoubleBox<T1,T2>::setContent( const T1 value1,const T2
value2)
    {
        a = value1;
        b = value2;
    }
int main(int argc, char **argv)
{
    int b=0;
    char p='a';
    float x=0.0;
    string t;
    DoubleBox <int,char> y;
    DoubleBox <float,string> v;
    v.setContent(3.2,"Check");
    y.setContent(7,'I');
    y.getContent(b,p);
    v.getContent(x,t);
    return 0;
}

```

Результат работы программы для двух экземпляров класса с разными атрибутами представлен на рисунке 3.

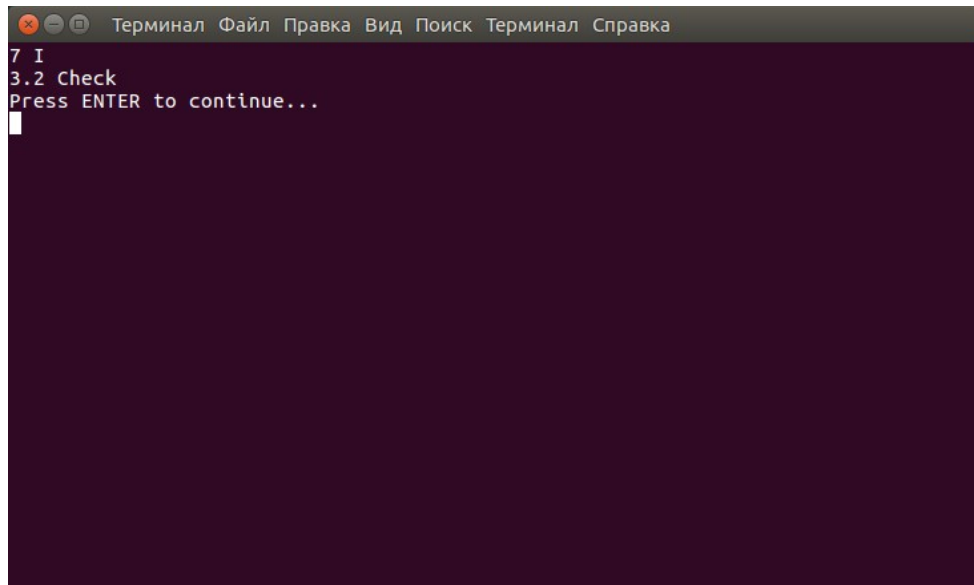


Рисунок 3 — Результат работы программы для двух экземпляров класса.

3.4 Был реализован шаблонный класс `Array` для хранения массива любого размера и типа, которые передаются в качестве параметров шаблона. В классе была реализована перегрузка оператора `[]` для обращения к элементам массива, конструктор по умолчанию, конструктор, позволяющий инициализировать весь массив одинаковыми значениями и конструктор, позволяющий инициализировать массив значениями внешнего массива.

Код программы:

```
#include <iostream>
using namespace std;
template <typename T, int size> class Array
{
private:
    T MAS[size];
public:
    Array() {}
    Array(T a) {
        for(int i=0; i<size; i++)
            MAS[i]=a;
    }
    Array(T*m,int s) {
        for(int i=0; i<s; i++)
            MAS[i]=m[i];
    }
    T& operator [](const int inde);
};
```

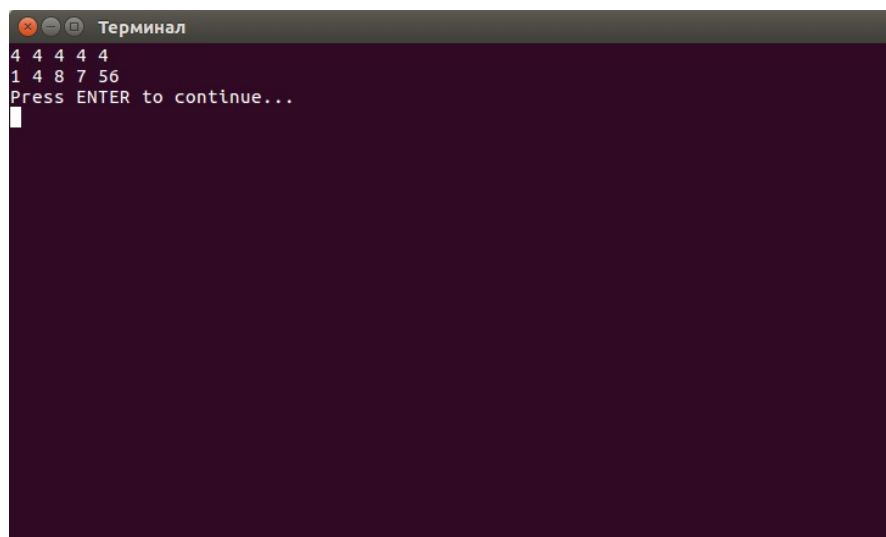


```

        const T& operator [] (const int inde) const;
};
template <typename T,int size>
T& Array<T,size>::operator [] (const int inde)
{
    return MAS[inde];
}
template <typename T,int size>
const T& Array<T,size>::operator [] (const int inde)
const
{
    return MAS[inde];
}
int main(int argc, char **argv)
{
    int massiv[4] {1,4,8,7};
    Array<char,10> z;
    Array <int, 5> x(4);
    Array <int,5> y(massiv,4);
    y[4]=56;
    for(int i=0;i<5;i++){
        cout<<x[i]<<" ";
    }
    cout<<endl;
    for(int i=0;i<5;i++){
        cout<<y[i]<<" ";
    }
    cout<<endl;
    return 0;
}

```

Результаты работы программы представлены на рисунке 4.



```
Терминал
4 4 4 4
1 4 8 7 56
Press ENTER to continue...

```

Рисунок 4 — Результат работы программы с шаблонным классом для хранения массива.

4 Вывод

В результате выполнения лабораторной работы были изучены возможности создания шаблонов в C++. Были освоена работа с шаблонными классами и функциями, получены практические навыки по методов и конструкторов шаблонных классов, шаблонных функций. Были получены практические навыки по перегрузке операторов для шаблонных классов.