**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И.РАЗЗАКОВА**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ**

**ОТЧЕТ**

**ПО ДИСЦИПЛИНЕ: ООП**

**ПО ТЕМЕ: Шаблоны**

**Лабораторная работа №7**

**ВЫПОЛНИЛ:** студент гр. ПИ(б)-3-18

Гадайбаев М.А.

**ПРОВЕРИЛА:** Каткова С.Н.

**Бишкек 2020**

**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ № 7**

1. Что такое обобщенная функция?

Шаблоны функций – это обобщенное описание поведения функций, которые могут вызываться для объектов разных типов.

1. Шаблоны функции позволяют удобным способом создать семейство

а) переменных;

**б) функций;**

в) классов;

г) программ.

1. Истинно ли утверждение, что шаблонная функция может иметь несколько аргументов?

В шаблоне функции можно использовать несколько шаблонных аргументов.

1. Реальный код шаблонной функции генерируется при:

а) объявлении функции в исходном коде;

б) создании объекта;

**в) запуске функции из её исходного кода.**

(Генерация кода при определении шаблона не происходит до тех пор,

пока функция не будет реально вызвана в ходе исполнения программы.)

1. Что такое обобщенный (шаблонный) класс?

Это класс, в которых тип данных передаётся, как формальный параметр.

1. В каком случае его имеет смысл применять?

* Избежание повторяемости написания программного кода для разных типов данных;
* Уменьшение текстовой части программного кода;
* Обеспечение удобного механизма передачи аргументов в шаблон класса с целью их обработки методами класса.

1. Шаблонный класс:

**a) работает с разными типами данных;**

б) генерирует идентичные объекты;

в) генерирует классы с различным числом методов

**Задание №1.**

**Постановка задачи:**

Создать шаблон функции, которая находит значение максимального элемента массива. Аргументами функции должны быть имя и размер массива (типа int). Создать шаблон функции, осуществляющей сортировку данных массива. Аргументами функции должны быть имя и размер массива (типа int). Продемонстрировать работу шаблонов на данных различных типов.

**Описание входных данных:**

arr1[] – массив типа int

arr2[] – массив типа double

arr3[] – массив типа char

arr4[] – массив типа float

size – размер массива типа int

max – максимальный элемент в массиве

MaxElem – обобщённая функция, которая находит значение максимального элемента массива.

BubbleSort – обобщённая функция, осуществляющая сортировку данных массива по возрастанию.

**Функция main() Рис.1.**

начало

MaxElem(arr1, size);

BubbleSort(arr1,size);

MaxElem(arr1, size);

BubbleSort(arr1,size);

MaxElem(arr1, size);

BubbleSort(arr1,size);

MaxElem(arr1, size);

BubbleSort(arr1,size);

конец

Рис.1.

**Обобщенная функция MaxElem Рис.2. и BubbleSort Рис.3.**

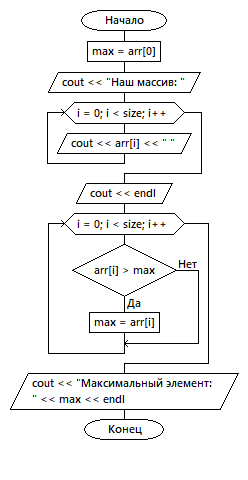
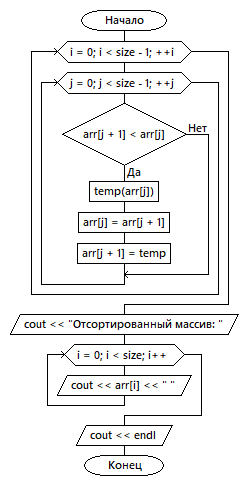
 

Рис.2. Рис.3.

**Код:**

#include<iostream>

using namespace std;

//поиск макс элемента

template <class T>

void MaxElem(T\* arr, int size) {

T max = arr[0];

cout << "Наш массив: ";

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

cout << endl;

for (int i = 0; i < size; i++) {

if (arr[i] > max) {

max = arr[i];

}

}

cout << "Максимальный элемент: "<< max << endl;

}

template <class T>

void BubbleSort(T\* arr, int size) {

//сортировка элементов

for (int i = 0; i < size - 1; ++i)

for (int j = 0; j < size - 1; ++j)

if (arr[j + 1] < arr[j]) {

T temp(arr[j]); //временная переменная должна быть того же типа, что и параметры

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

cout << "Отсортированный массив: ";

for (int i = 0; i < size; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

setlocale(LC\_ALL, "RUS");

const int size = 5;

int arr1[size] = { 10, 5, 7, 3, 4};

double arr2[size] = { 7.7682, 2.234, 4.67876, 1.2345, 5.346 };

char arr3[size] = { 'f', 'a', 'c', 'h', 'z' };

float arr4[size] = { 7.62 , 5.56 , 38.0 , 56.0 , 9.0 };

//вызов шаблона функции

//вызов функции с аргументами типа int

MaxElem(arr1, size);

BubbleSort(arr1, size);

cout << endl;

//вызов функции с аргументами типа double

MaxElem(arr2, size);

BubbleSort(arr2, size);

cout << endl;

//вызов функции с аргументами типа char

MaxElem(arr3, size);

BubbleSort(arr3, size);

cout << endl;

//вызов функции с аргументами типа float

MaxElem(arr4, size);

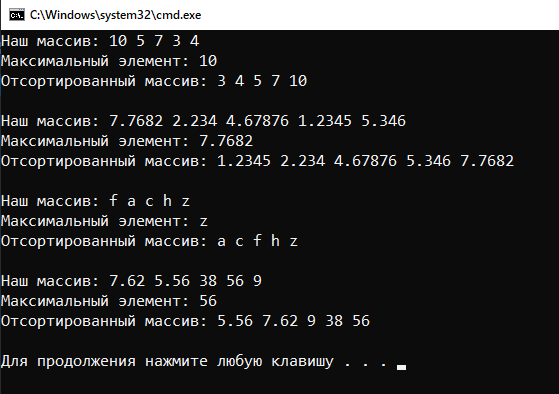
BubbleSort(arr4, size);

cout << endl;

return 0;

}

**Результат:**

****

**Задание №2.**

**Постановка задачи:**

Создать шаблон класса, у которого есть 2 поля в виде одномерных массивов одного размера. Заполнить массивы с помощью генератора случайных чисел. Найти сумму элементов этих массивов. Продемонстрировать работу шаблонов на данных различных типов.

**Описание входных данных:**

Класс SumArray – обобщенный класс, с полями: size – размер массива; firstArr[] – первый массив; secondArr[] – второй массив, методы: конструктор с аргументами; деструктор; randArr() – для заполнения двух массивов с помощью генератора случайных чисел; print() – вывод на консоль первого и второго массива; sum() – сложение двух массивов поэлементно.

Класс SumArray<char> - явная специализация обобщенного класса для char. Поля и методы, идентичные с обобщенным классом, кроме метода randArr() – заполнение массива рандомными **символами**.

**SumArray<int> obj1** – объект обобщённого класса **SumArray** c полем int. Переменные объекта: size – размер массива;

**SumArray<double> obj2** - объект обобщённого класса **SumArray** c полем **double**. Переменные объекта: size – размер массива;

**SumArray<char> obj3** - объект явно специализированного класса **SumArray** c полем **char**. Переменные объекта: size – размер массива;

**Класс SumArray Рис.1. Функция main Рис.2.**

Рис.1. Рис.2.

**Метод randArr() Рис.3. Метод print() Рис.4.**

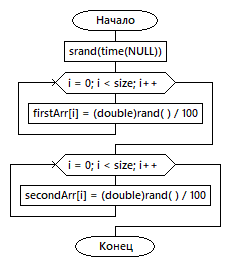
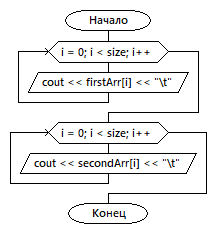
 

Рис.3. Рис.4.

**Метод sum() Рис.5.**

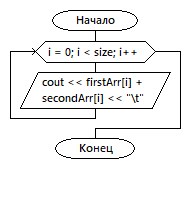
****

Рис.5.

**Код:**

#include<iostream>

#include<ctime>

#include<cstdlib>

using namespace std;

//шаблон класса SumArray

template <class T>

class SumArray

{

private:

//поля класса

int size;

T firstArr[INT16\_MAX];

T secondArr[INT16\_MAX];

public:

//методы класса

//конструктор с аргументами, передаем только размер массива, т.к. сам массив заполняем здесь

SumArray(int s) {

this -> size = s;

}

//деструктор

~SumArray() {

cout << "очищено!" << endl;

}

//заполнение массива рандомными числами

void randArr() {

srand(time(NULL));

for (int i = 0; i < size; i++) {

firstArr[i] = (double)rand() / 100;

}

for (int i = 0; i < size; i++) {

secondArr[i] = (double)rand() / 100;

}

}

//вывод на консоль первого и второго массива

void print() {

for (int i = 0; i < size; i++) {

cout << "firstArr[" << i << "] = " << firstArr[i] << "\t";

}

cout << endl;

for (int i = 0; i < size; i++) {

cout << "secondArr[" << i << "] = " << secondArr[i] << "\t";

}

cout << endl<<endl;

}

//сложение двух массивов поэлементно

void sum() {

for (int i = 0; i < size; i++) {

cout << "Arr[" << i << "] = " << firstArr[i] + secondArr[i] << "\t";

}

cout << endl << endl;

}

};

// явная специализация обощенного класса для char

template <>

class SumArray<char>

{

private:

//поля класса

int size;

char firstArr[INT16\_MAX];

char secondArr[INT16\_MAX];

public:

//методы класса

//конструктор с аргументами, передаем только размер массива, т.к. сам массив заполняем здесь

SumArray(int s) {

this->size = s;

}

//деструктор

~SumArray() {

cout << "очищено!" << endl;

}

//заполнение массива рандомными числами

void randArr() {

srand(time(NULL));

for (int i = 0; i < size; i++)

firstArr[i] = char('a' + rand() % ('z' - 'a'));

for (int i = 0; i < size; i++)

secondArr[i] = char('a' + rand() % ('z' - 'a'));

}

//вывод на консоль первого и второго массива

void print() {

for (int i = 0; i < size; i++) {

cout << "secondArr[" << i << "] = " << secondArr[i] << "\t";

}

cout << endl << endl;

for (int i = 0; i < size; i++) {

cout << "firstArr[" << i << "] = " << firstArr[i] << "\t";

}

cout << endl << endl;

}

//сложение двух массивов поэлементно

void sum() {

for (int i = 0; i < size; i++) {

cout << "Arr[" << i << "] = " << firstArr[i] + secondArr[i] << "\t";

}

cout << endl << endl;

}

};

int main() {

setlocale(LC\_ALL, "RUS");

//int size;

//cout << "Введите size: ";

//cin >> size;

SumArray<int> obj1(5);

cout << "Массивы типа int:" << endl;

obj1.randArr();

obj1.print();

cout << "Сумма массивов:" << endl;

obj1.sum();

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

SumArray<double> obj2(5);

cout << "Массивы типа double:" << endl;

obj2.randArr();

obj2.print();

cout << "Сумма массивов:" << endl;

obj2.sum();

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

SumArray<char> obj3(5);

cout << "Массивы типа char:" << endl;

obj3.randArr();

obj3.print();

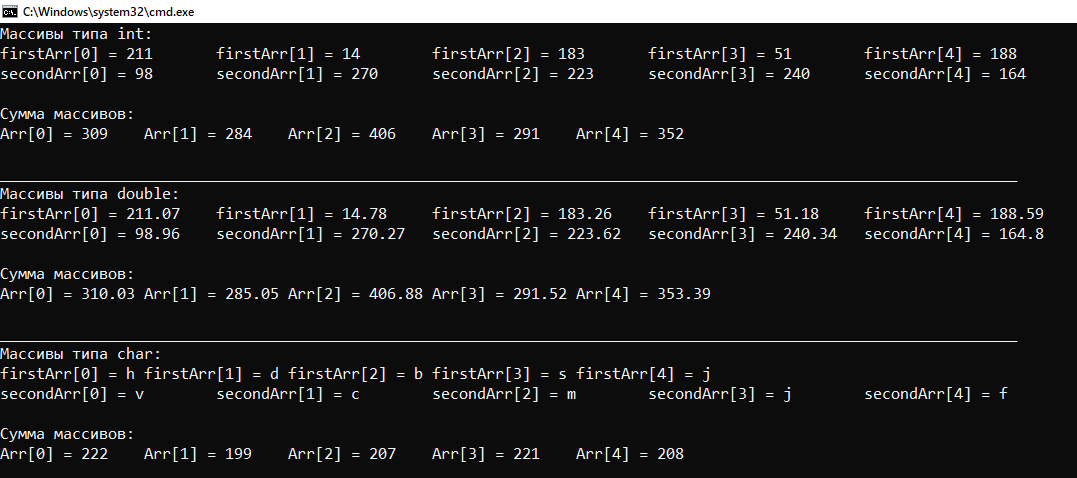
cout << "Сумма массивов:" << endl;

obj3.sum();

return 0;

}

**Результат:**



**Задание №3.**

**Постановка задачи:**

Создать шаблон класса «стек». Использовать его при решении задачи 3 из лабораторной № 1.

**Описание входных данных:**

Класс Stack – обобщенный класс

с полями: item – элемент стека; \*next – указатель на следующий элемент; \*top – голова стека,

методы: set() – установка значения полей; push() – ввод элементов в стек;

printStack() – вывод стека на экран; pop() – удаление элемента из стека; getStackSize() – получить размер стека; isEmpty() – проверка на пустоту; infile() – чтение данных из файла; split() – разделение данных на два вагона.

**Stack<int> TrainBase1, OneTrain1, TwoTrain1** – объекты обобщённого класса Stack c полем int.

**Stack<double> TrainBase2, OneTrain2, TwoTrain2** – объекты обобщённого класса Stack c полем double.

**Stack<char> TrainBase3, OneTrain3, TwoTrain3** – объекты обобщённого класса Stack c полем char.

**Класс Stack Рис.1.**



Рис.1.

**Функция main (для типа integer) Рис.2.**



Рис.2.

**Метод push() Рис.3.**

**Метод PrintStack() Рис.4.**

**Метод getStackSize() Рис.5.**

**Метод pop() Рис.6.**



Рис.3. Рис.4. Рис.5. Рис.6.

**Метод infile() Рис.7.**



Рис.7.

**Код:**

#include <iostream>

#include <conio.h>

#include <vector>

#include <iomanip>

#include <string>

#include <Windows.h>

#include <fstream>

using namespace std;

//шаблон класса SumArray

template <class T>

class Stack {

private:

//поля класса

struct {

T item; //элемент стека

Stack\* next; //указатель на next

};

Stack\* top = NULL;

public:

//методы класса

void set(T item) {

this->item = item;

}

void push() { //ВВОД ЭЛЕМЕНТОВ

Stack\* pv = new Stack;

pv->item = item;

pv->next = top;

top = pv;

}

void printStack() { // ВЫВОДИМ СТЕК НА ЭКРАН

if (isEmpty()) {

cout << "вагон пуст";

}

Stack\* pv = top;

while (pv != NULL) {

cout << pv->item << " ";

pv = pv->next;

}

cout << "\n";

delete pv;

}

T pop() { //УДАЛЕНИЕ ЭЛЕМЕНТА

if (isEmpty()) {

cout << "стек пуст" << endl;

}

else {

T temp = top->item;

Stack\* pv = top;

top = top->next;

delete pv;

return temp;

}

}

void getStackSize() { //ПОЛУЧИТЬ РАЗМЕР СТЕКА

int sz = 0;

Stack\* pv = top;

while (pv != NULL)

{

sz++;

pv = pv->next;

}

cout << sz << "\n";

delete pv;

}

bool isEmpty() //ПРОВЕРКА НА ПУСТОТУ

{

return top ? false : true;

}

void infile(string source) { //ЧТЕНИЕ ДАННЫХ ИЗ ФАЙЛА

ifstream file(source+".txt");

if (file) {

while (!file.eof()) {

T elem;

file >> elem;

set(elem);

push();

}

}

}

T split(T a, T b) { //РАЗДЕЛЕНИЕ ДАННЫХ НА ДВА ВАГОНА

T elem = pop();

if (elem == a) {

return a;

}

else if (elem == b) {

return b;

}

}

};

int main() { //ГЛАВНАЯ МЕНЮШКА, УПРАВЛЯЕТ ПОСЛЕДОВАТЕЛЬНОСТЬЮ ВЫЗОВА ВСЕХ МЕТОДОВ КЛАССА

setlocale(LC\_ALL, "rus");

// работаем с типом integer

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_INTEGER\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Stack<int> TrainBase1, OneTrain1, TwoTrain1;

int a1=0, b1=1;

string source1 = "integer";

TrainBase1.infile(source1);

cout << "Основной вагон: ";

TrainBase1.printStack();

cout << "Количество элементов: ";

TrainBase1.getStackSize();

while (!TrainBase1.isEmpty()) {

int temp1 = TrainBase1.split(a1, b1);

if (temp1 == a1) {

OneTrain1.set(temp1); OneTrain1.push();

}

else{

TwoTrain1.set(temp1); TwoTrain1.push();

}

}

cout << endl;

cout << "Сортировка по вагонам:" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Основной вагон: ";

TrainBase1.printStack();

cout << "Первый вагон: ";

OneTrain1.printStack();

cout << "Второй вагон: ";

TwoTrain1.printStack();

cout << endl << endl;

//работаем с типом double

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DOUBLE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Stack<double> TrainBase2, OneTrain2, TwoTrain2;

double a2 = 1.2, b2 = 1.3;

string source2 = "double";

TrainBase2.infile(source2);

cout << "Основной вагон: ";

TrainBase2.printStack();

cout << "Количество элементов: ";

TrainBase2.getStackSize();

while (!TrainBase2.isEmpty()) {

double temp2 = TrainBase2.split(a2, b2);

if (temp2 == a2) {

OneTrain2.set(temp2); OneTrain2.push();

}

else {

TwoTrain2.set(temp2); TwoTrain2.push();

}

}

cout << endl;

cout << "Сортировка по вагонам:" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Основной вагон: ";

TrainBase2.printStack();

cout << "Первый вагон: ";

OneTrain2.printStack();

cout << "Второй вагон: ";

TwoTrain2.printStack();

cout << endl << endl;

//работаем с типом char

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CHAR\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Stack<char> TrainBase3, OneTrain3, TwoTrain3;

char a3 = 'a', b3 = 'b';

string source3 = "char";

TrainBase3.infile(source3);

cout << "Основной вагон: ";

TrainBase3.printStack();

cout << "Количество элементов: ";

TrainBase3.getStackSize();

while (!TrainBase3.isEmpty()) {

char temp3 = TrainBase3.split(a3, b3);

if (temp3 == a3) {

OneTrain3.set(temp3); OneTrain3.push();

}

else {

TwoTrain3.set(temp3); TwoTrain3.push();

}

}

cout << endl;

cout << "Сортировка по вагонам:" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Основной вагон: ";

TrainBase3.printStack();

cout << "Первый вагон: ";

OneTrain3.printStack();

cout << "Второй вагон: ";

TwoTrain3.printStack();

cout << endl << endl;

system("pause>NULL");

return 0;

}

**Результат:**

