

Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Факультет «О» Естественнаучный

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №3**  
по дисциплине «Программирование на ЯВУ»  
на тему «КЛАССЫ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ»

Выполнил:  
Студент Костров Г. Ю.  
Группа О712Б  
Преподаватель: Васюков В.М.

Санкт-Петербург  
2022 г

## Постановка задачи:

Написать шаблон функции, выполняющей указанные в вариативной части задания действия. Написать программу тестирования шаблонных функций, созданных на основе этого шаблона, с аргументами указанных типов. Разработать шаблон класса, описывающий указанный в вариативной части задания абстрактный тип данных, и написать программу тестирования объектов двух шаблонных классов. Выбор тестируемого метода должен осуществляться с помощью меню. Это задание может быть выполнено на трех уровнях сложности:

**Повышенный.** Создать требуемый АД с помощью двух структур хранения: векторной и списковой, реализацию оформить в виде шаблонов классов с единым интерфейсом.

### Вариант 12

Типы аргументов: int, char

1. Перестановка элементов в массиве следующим образом: сначала отрицательные элементы в порядке убывания, затем неотрицательные в порядке возрастания.
2. АД Стек. Структура хранения – связанный список.

## Текст программы:

### // testFunction.h

```
#if !defined(TestFunction_H)
#define TestFunction_H

// Заголовочные файлы
#include "iostream"
#include "algorithm"
#include "string"
// Функции-помощники
#include "..\helpFunctions\functions.h"

using namespace std;

// Функции
template <class T>
void testFunctionInput(int size, int max, T *list);

template <class T>
int printSortedList (T mas[], int n);

template <class T>
void sort (T mas[], int n, int flag);

int getVariantTemplate(int count);

#endif // TestFunction_H
```

### // testFunction.cpp

```
#include "testFunction.h"

// Сортировка по возрастанию
template <class T>
void sort (T mas[], int n, int flag) {
```

```

    for (int startIndex = 0; startIndex < n - 1; ++startIndex) {
        int smallestIndex = startIndex;

        for (int currentIndex = startIndex + 1; currentIndex < n; ++currentIndex)
        {
            if (flag == 1) { // в зависимости от флага сортируем по убыванию или
возрастанию
                if (mas[currentIndex] < mas[smallestIndex])
                    smallestIndex = currentIndex;
            } else {
                if (mas[currentIndex] > mas[smallestIndex])
                    smallestIndex = currentIndex;
            }
        }
        std::swap(mas[startIndex], mas[smallestIndex]);
    }
}

```

```

template <class T>
int printSortedList (T mas[], int n) {

    system("cls"); // очищаем экран
    cout << endl << "Start list" << endl;
    for (int index = 0; index < n; ++index) cout << (int) mas[index] << " ";
    cout << endl;
    sort(mas, n, 1);

    int temp;
    // поиск первого элемента, что больше нуля
    for (int startIndex = 0; startIndex < n ; ++startIndex) {
        if (mas[startIndex] > 0) {
            temp = startIndex;
            break;
        }
    }

    sort(mas, temp, 0);

    // вывод значений массива
    cout << endl << "Sorted list" << endl;
    for (int index = 0; index < n; ++index) cout << (int) mas[index] << " ";

    return n;
}

```

```

void testFunction() {
    system("cls");
    // ввод размера массива
    cout << "Enter size of array(0 < Your number <= 10)" << endl << ">";
    int size = getVariant(10);

    // выбор типа массива
    string menu[] = {
        "Choose testing args type",
        "1. int",
        "2. char"};
    printMenu(menu, 3);
}

```

```

int type = getVariant(2);

cout << endl << "Enter " << size << " Numbers" << endl ;
if (type == 1) {
    int list[size];
    testFunctionInput(size, 1000000000, list);
} else {
    char list[size];
    testFunctionInput(size, 128, list);
}
cout << endl;
system("pause");

}

// Input and mas sorting
template<class T>
void testFunctionInput(int size, int type, T *list) {
    for (int i = 0; i < size; i++) {
        cout << "list[" << i << "] = ";
        list[i] = getVariantTemplate(type);
    }
    printSortedList(list, size);
}

// Умный input элементов
int getVariantTemplate(int count) {
    int var;
    cin.clear();
    string s; // строка для считывания введённых данных
    getline(cin, s); // считываем строку
    // пока ввод некорректен, сообщаем об этом и просим повторить его
    while (sscanf(s.c_str(), "%d", &var) != 1 || var < (-count+1) || var > count)
    {
        if (s.size() != 0) {
            cout << "Incorrect input. Try again: "; // выводим сообщение об ошибке
            getline(cin, s); // считываем строку повторно
        } else {
            var = getVariantTemplate(count);
            if (var >= 1 || var <= count) {
                break;
            }
        }
    }

    return var;
}

// testClasses.h

#ifndef TEST_CLASSES_H
#define TEST_CLASSES_H

// Заголовочные файлы
#include "string"
// функции-помощники
#include "../helpFunctions/functions.h"
// файлы проекта
#include "../Classes/stackArray.h"

```

```

#include "..\Classes\stackList.h"
#include "..\Classes\stack.h"

using namespace std;

template <class T>
void templateClassTest (Stack<T> * );

#endif // TEST_CLASSES_H

```

## // testClasses.h

```

#include "testClasses.h"

void testClasses() {
    int variant;

    do {
        system("cls"); // очищаем экран

        string menu[] = {
            "Select structures storage",
            "1. Linked list",
            "2. Vector"};
        printMenu(menu, 3);
        exitMenu(3);

        variant = getVariant(3);

        switch (variant) {
            case 1: {
                List<int> *stek;
                stek = new List<int>;
                templateClassTest(stek);

                break;
            }

            case 2: {
                Array<int> *stek;
                stek = new Array<int>(10);
                templateClassTest(stek);

                break;
            }

            default:
                break;
        }

    } while (variant != 3);
}

```

```

template<class T>
void templateClassTest (Stack<T>* expamleMas) {

```

```

int variant;

do {
    system("cls");

    string menu[] = {
        "What do you want to do?",
        "1. Push",
        "2. Pop",
        "3. Top",
        "4. Empty",
        "5. Full"};
    printMenu(menu, 6);
    exitMenu(6);

    variant = getVariant(6);

    switch (variant) {
        case 1: {
            cout << "Enter value" << endl << ">";
            int value = getVariant(100);
            expamleMas->Push(value);
            break;
        }

        case 2: {
            expamleMas->Pop();
            system("pause");
            break;
        }

        case 3: {
            T temp = expamleMas->Top();
            if (temp != -1) {
                cout << "Top value is " << temp << endl;
            }
            system("pause");
            break;
        }

        case 4: {
            if (expamleMas->Empty()) {
                cout << "Stack is empty :(" << endl;
            } else {
                cout << "Stack is not empty :)" << endl;
            }
            system("pause");
            break;
        }

        case 5: {
            if (expamleMas->Full()) {
                cout << "Stack is full :(" << endl;
            } else {
                cout << "Stack is not full :)" << endl;
            }
            system("pause");
            break;
        }
    }
}

```

```

    }

    }while(variant != 6);
}

```

## // main.h

```

#ifndef MAIN_H
#define MAIN_H

// функции-помощники
#include "..\helpFunctions\functions.cpp"
// заголовочные файлы
#include "string"
// Заголовочные файлы программы
#include "testFunction.h"

using namespace std;

// Тест функции
void testFunction();

// Тест класса
void testClasses();

#endif // main.h

```

## // main.cpp

```

#include "main.h"

int main() {

    int variant = 0;
    do {

        system("cls"); // очищаем экран

        string menu[] = {
            "What do you want to do?",
            "1. Test function template",
            "2. Test class template"};
        printMenu(menu, 3);
        exitMenu(3);

        variant = getVariant(3);

        switch (variant) {
            case 1: {
                testFunction();
                break;
            }
        }
    }
}

```

```

        case 2: {
            testClasses();
            break;
        }

        case 3:
            break;
    }

    while (variant != 3);

    return 0;
}

```

## Результат работы программы:

Рисунок 1: Основное меню.

Рисунок 2: Тестирование шаблона функции(ввод значений).

Рисунок 3: Тестирование шаблона функции(вывод отсортированных значений).

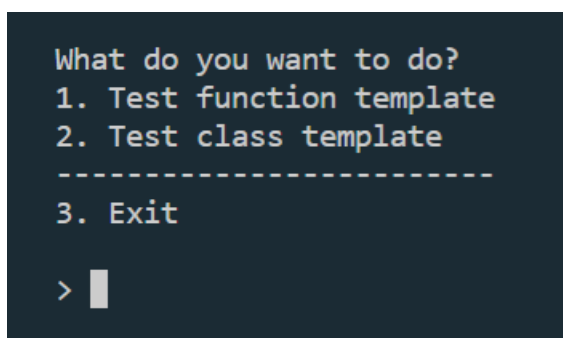
Рисунок 4: Тестирование шаблона класса(выбор структуры хранения).

Рисунок 5: Тестирование шаблона класса(добавление элемента в стек).

Рисунок 6: Тестирование шаблона класса(неразрушающее чтение элемента с вершины стека.).

Рисунок 7: Тестирование шаблона класса(проверка пустоты стека).

Рисунок 8: Тестирование шаблона класса(проверка заполнения стека).



```

What do you want to do?
1. Test function template
2. Test class template
-----
3. Exit

> █

```

Рисунок 1



```

Enter size of array(0 < Your number <= 10)
>5
Choose testing args type
1. int
2. char
1

Enter 5 Numbers
list[0] = 3
list[1] = 2
list[2] = 4
list[3] = 5
list[4] = 3

```

Рисунок 2

```

Start list
3 2 4 5 3

Sorted list
2 3 3 4 5
Для продолжения нажмите любую клавишу . . .

```

Рисунок 3

```

Select structures storage
1. Linked list
2. Vector
-----
3. Exit

>

```

Рисунок 4

```

What do you want to do?
1. Push
2. Pop
3. Top
4. Empty
5. Full
-----
6. Exit

> 1
Enter value
>45

```

Рисунок 5

```
What do you want to do?
1. Push
2. Pop
3. Top
4. Empty
5. Full
-----
6. Exit

> 3
Top value is 5
Для продолжения нажмите любую клавишу . . . █
```

Рисунок 6

```
What do you want to do?
1. Push
2. Pop
3. Top
4. Empty
5. Full
-----
6. Exit

> 4
Stack is not empty :)
Для продолжения нажмите любую клавишу . . . █
```

Рисунок 7

```
What do you want to do?
1. Push
2. Pop
3. Top
4. Empty
5. Full
-----
6. Exit

> 5
Stack is not full :)
Для продолжения нажмите любую клавишу . . . █
```

Рисунок 8

