1. Постановка задачи

Важно учитывать при реализации:

● Необходимо выполнить разделение на h и cpp файлы для каждого класса. h файлы содержат определение, cpp файлы содержат реализацию. функция main обязана располагаться в отдельном cpp файле.

● Реализовать динамическое выделение памяти и очищение.

● Данные класса обязаны находиться в области доступа private. Должны быть созданы функции для получения значения данных и установки значения. Прямого доступа к данным быть не должно.

● Если требуется реализовать стек/очередь/дек/список, то элемент такой дисциплины обязан быть выполнен в виде класса. В классе элемента данные и ссылка (-и) обязаны располагаться в области private. Обязательно создание функций для извлечения данных и изменения этих данных и ссылок (-и)..

● Обязательно реализовать работу с исключительными ситуациями - генерация и обработку. Если в ЛР студенту не очевидно какую исключительную ситуацию следует обрабатывать, следует обратиться к преподавателю. Обработка исключений должна производиться в обоих заданиях.

● Реализовать пользовательское меню согласно заданию. Обязательно реализовать возможность выбора типа данных, с которыми возможно взаимодействие в каждом задании: int, char, float, double, char*. Не должно быть в программе параметров, которые задаются в main, все, что может задать пользователь должно задаваться с клавиатуры, если в задании не указано иначе (имеется в виду заполнение случайными данными).

● Класс должен содержать конструктор со списком инициализации, конструктор с параметром, деструктор.

● По списку студент определяет свой порядковый номер. Необходимо выполнить оба задания.

Задание. Вариант №3:

Задание 1

Написать функцию-шаблон, переставляющую элементы в массиве с шагом, заданным пользователем. Размер и значения массива генерируются случайным образом (до 20).

Задание 2

Создать параметризованный массив с перегруженными операторами [] для доступа поиндексу, = для присваивания массивов друг другу, вывода и ввода в поток, == для сравнения массивов.

2. Формализация

2.1 Формализация первого задания

Функция шаблон будет располагаться в одном файле с классом параметризованного массива. В main для тестирования этой функции будет создано 5 массивов разных типов на куче, указатели на которые будут передаваться в функцию.

2.2 Формализация второго задания

Стоит отметить, что условие разделения на h и cpp файлы выполнимо только в случае, когда подключается не h файл, а cpp файл, так как линковщик не может обнаружить шаблонный тип из-за природы его создания. Правильней будет реализовать весь функционал в h файле.

3. Исходный код заданий №1 и №2

lab3.h

```
//
// Created by user on 4/18/2024.
//
```

```
#ifndef CMAKEPROJECT_LAB4_H
#define CMAKEPROJECT_LAB4_H


#include "../libs/libs.h"
using namespace std;
int lab4();
#include "TempArray.h"


#endif //CMAKEPROJECT_LAB4_H
```

lab3.cpp

```
//
// Created by user on 4/18/2024.
//

#include "lab4.h"
int lab4() {
    system("cls");
    int inp,type;
    cout<<"Please choose type ypu want to work:"<<endl;
    cout<<"1. int"<<endl;
    cout<<"2. char"<<endl;
    cout<<"3. float"<<endl;
    cout<<"4. double"<<endl;
    cout<<"5. string"<<endl;
    cin>>type;
    if(type<1 || type>5) {
        cout<<"Try again and choose correct type!"<<endl;
        return -1;
    }

    TempArray<int> i1;
    TempArray<char> c1;
    TempArray<float> f1;
    TempArray<double> d1;
    TempArray<string> s1;

    int* i2;
    char* c2;
    float* f2;
    double* d2;
    string* s2;

    srand(time(nullptr));
```

```
    unsigned len = 1+rand()%20;
    i2 = new int[len];
    c2 = new char [len];
    f2 = new float [len];
    d2 = new double [len];
    s2 = new string [len];
    for(unsigned i = 0;i<len;++i){
        i2[i] = rand()%101-50;
         c2[i] = rand()%101-50;
         f2[i] = rand()%101-50;
         d2[i] = rand()%101-50;
         s2[i] = to_string(rand()%101-50);
    }

    system("cls");
    do{
        cout<<"1.  Create(Replace)  TempArray  of  length  and
enter it"<<endl;
        cout<<"2. Print TempArray"<<endl;
        cout<<"3. Print TempArray[index]"<<endl;
        cout<<"4. Print array"<<endl;
        cout<<"5. Hard swap with step and swap step"<<endl;
        cout<<"6. End"<<endl;

        cin>>inp;
        system("cls");

        switch(inp){
            case 1:
                unsigned length;
                cout<<"Enter length: ";
                cin>>length;
                switch(type){
                    case 1:{
                        TempArray<int> i3{length};
                        i3.operator>>(cin);
                        i1 = i3;
                        break;
                    }
                    case 2:{
                        TempArray<char> c3{length};
                        c3.operator>>(cin);
                        c1 = c3;
                        break;
                    }
```

```
            case 3:{
                TempArray<float> f3{length};
                f3.operator>>(cin);
                f1 = f3;
                break;
            }
            case 4:{
                TempArray<double> d3{length};
                d3.operator>>(cin);
                d1 = d3;
                break;
            }
            case 5:{
                TempArray<string> s3{length};
                s3.operator>>(cin);
                s1 = s3;
                break;
            }
            default:
                break;
        }
        break;
    case 2:
        switch(type){
            case 1:
                i1.operator<<(cout)<<endl;
                break;
            case 2:
                c1.operator<<(cout)<<endl;
                break;
            case 3:
                f1.operator<<(cout)<<endl;
                break;
            case 4:
                d1.operator<<(cout)<<endl;
                break;
            case 5:
                s1.operator<<(cout)<<endl;
                break;
            default:
                break;
        }
        break;
    case 3:
        int c;
```

```
                    cout<<"Enter index: ";
                    cin>>c;
                    try {
                        switch(type){
                            case 1:
                                cout<<i1[c]<<endl;
                                break;
                            case 2:
                                cout<<c1[c]<<endl;
                                break;
                            case 3:
                                cout<<f1[c]<<endl;
                                break;
                            case 4:
                                cout<<d1[c]<<endl;
                                break;
                            case 5:
                                cout<<s1[c]<<endl;
                                break;
                            default:
                                break;
                        }
                    }catch (const char * error){
                        cerr<<error;
                        delete[] i2;
                        delete[] c2;
                        delete[] f2;
                        delete[] d2;
                        return -1;
                    }
                    break;
                case 4:
                    switch (type) {
                        case 1:
                            for (int i = 0; i < len; ++i) cout <<
i2[i] << " ";
                            cout << endl;
                            break;
                        case 2:
                            for (int i = 0; i < len; ++i) cout <<
c2[i] << " ";
                            cout << endl;
                            break;
                        case 3:
```

```
                        for (int i = 0; i < len; ++i) cout <<
f2[i] << " ";
                        cout << endl;
                        break;
                    case 4:
                        for (int i = 0; i < len; ++i) cout <<
d2[i] << " ";
                        cout << endl;
                        break;
                    case 5:
                        for (int i = 0; i < len; ++i) cout <<
s2[i] << " ";
                        cout << endl;
                        break;
                    default:
                        break;
                }
                break;
            case 5:
                int step,swap;
                cout<<"Enter step: ";
                cin>>step;
                cout<<"Enter swap: ";
                cin>>swap;
                try{
                    switch(type){
                        case 1:
                            hardSwap(i2,len,step,swap);
                            break;
                        case 2:
                            hardSwap(c2,len,step,swap);
                            break;
                        case 3:
                            hardSwap(f2,len,step,swap);
                            break;
                        case 4:
                            hardSwap(d2,len,step,swap);
                            break;
                        case 5:
                            hardSwap(s2,len,step,swap);
                            break;
                        default:
                            break;
                    }
                }catch (const char* error){
```

```
                        cerr<<error;
                        delete[] i2;
                        delete[] c2;
                        delete[] f2;
                        delete[] d2;
                        return -1;
                    }
                    break;
                case 6:
                    break;
                default:
                    if (inp == EOF) inp = 7;
                    cout << "There is no enough command. If you
    want to exit enter 6\n";
                    break;
            }
        }while(inp != 6);

        delete[] i2;
        delete[] c2;
        delete[] f2;
        delete[] d2;

        return 0;
    }
```

TempArray.h

```
    //
    // Created by user on 4/18/2024.
    //

    #ifndef CMAKEPROJECT_TEMPARRAY_H
    #define CMAKEPROJECT_TEMPARRAY_H

    #include <iostream>



    template<typename T>
    class TempArray {

    public:

        explicit TempArray(unsigned length = 10);
        ~TempArray();
```

```
    unsigned getLen();
    T& operator [] (unsigned index);
    TempArray<T>& operator = (const TempArray<T>&);

    std::ostream& operator <<(std::ostream&);
    std::istream& operator >>(std::istream&);
    bool operator == (const TempArray<T>&);
private:
    unsigned len;
    T* array;
};

template<typename T>
void hardSwap(T* array, unsigned len,
              unsigned step = 1, unsigned swapStep = 1) {
    if(swapStep>=len){
        throw "Swap step > len of array!\n";
    }
    T tmp;
    for(unsigned i = 0; i<len;i+=step){
        if(i+swapStep>=len) break;
        tmp = array[i];
        array[i] = array[i+swapStep];
        array[i+swapStep] = tmp;
    }

}

template<typename T>
TempArray<T>::TempArray(unsigned length) :
        len(length>0?length:10),
        array(new T[length>0?length:10]){}


template<typename T>
TempArray<T>::~TempArray() {
    delete[] array;
}

template<typename T>
T &TempArray<T>::operator[](unsigned int index) {
    if(index >= len) throw "Index out of range!\n";
    else return array[index];
}
```

```
template<typename T>
TempArray<T>    &TempArray<T>::operator=(const    TempArray<T>
&another) {
    delete[] array;
    len = another.len;
    array = new T[len];
    unsigned s = 0;
    while(s != len){
        array[s] = another.array[s];
        s++;
    }
    return *this;
}



template<typename T>
std::ostream& TempArray<T>::operator<<(std::ostream &os) {
    unsigned s = 0;
    while(s<len){
        os<<array[s]<<" ";
        s++;
    }
    return os;
}

template<typename T>
std::istream& TempArray<T>::operator>>(std::istream &is) {
    unsigned s = 0;
    while(s<len){
        is>>array[s];
        s++;
    }
    return is;
}

template<typename T>
bool TempArray<T>::operator==(const TempArray<T> &another) {
    if(len!=another.len) return false;
    unsigned s = 0;
    while(s<len){
        if(array[s]!=another.array[s]) return false;
        s++;
    }
    return true;
```

```
}

template<typename T>
unsigned TempArray<T>::getLen() {
    return len;
}



#endif //CMAKEPROJECT_TEMPARRAY_H
```

4. Результаты работы программы

При начале работы программы дается выбор с каким типом данных работать.

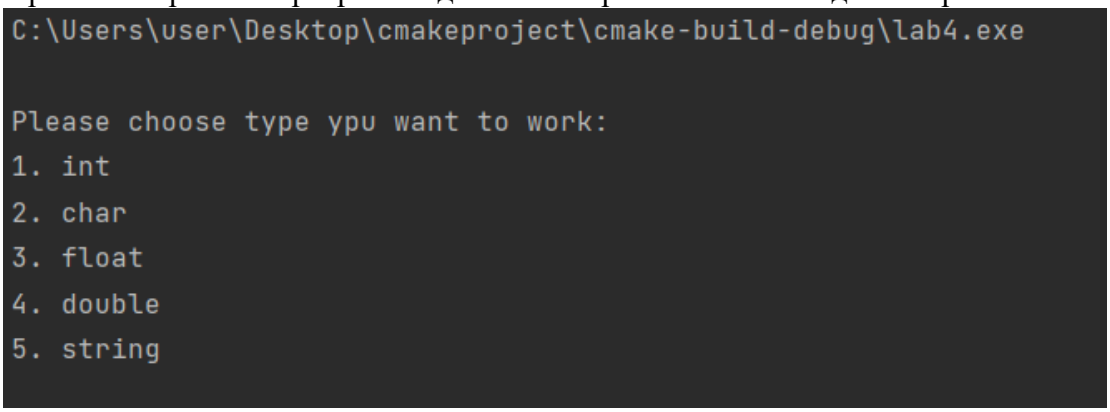

Рис. 1 – Выбор типа данных

Выбрать повторно можно завершив работу программы и перезапустив ее. После выбора перед пользователем появляется главное меню.

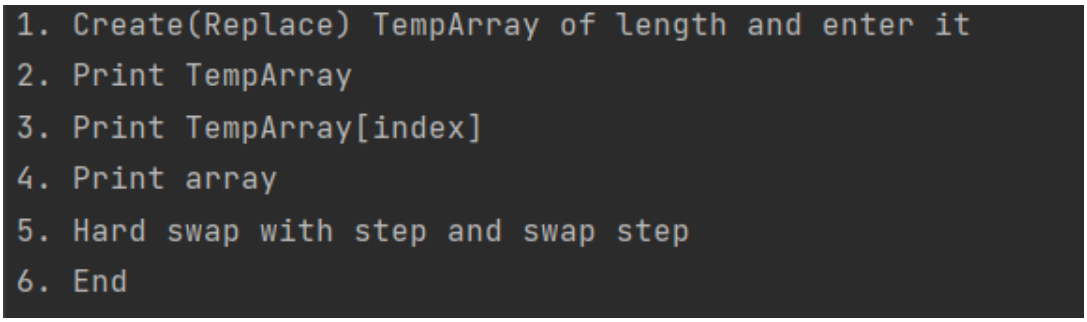

Рис. 2 – Главное меню

4.1 Результаты работы задания №2

Первые три пункта работают с параметризированным массивом, реализованным как шаблонный класс TempArray. Первая команда запрашивает длину массива и просит ввести его.

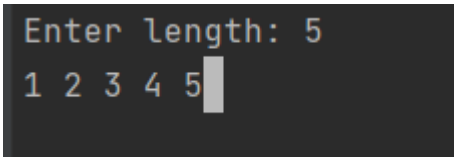

Рис. 3 – Ввод параметризованного массива

Вторая команда выводит введенный массив

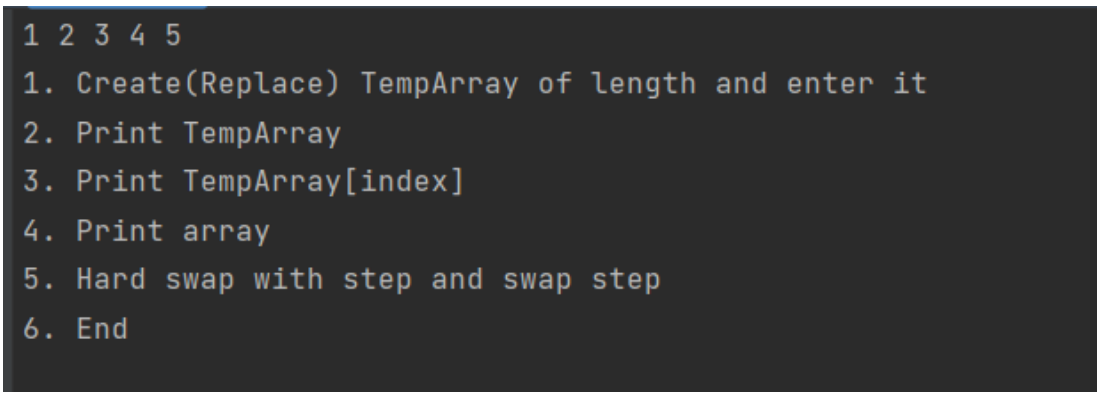

Рис. 4 – Вывод параметризированного массива

Третья команда запрашивает индекс и выводит элемент по этому индексу.

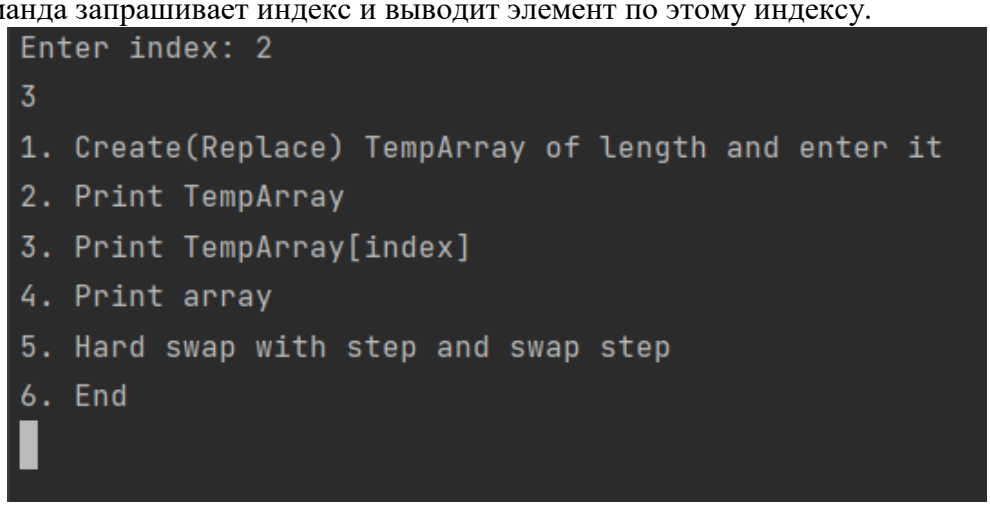

Рис. 5 – Вывод элемента параметризированного массива

Также существует обработка ошибок. В случае ввода неверного индекса программа выдаст ошибку и аварийно завершит свою работу.

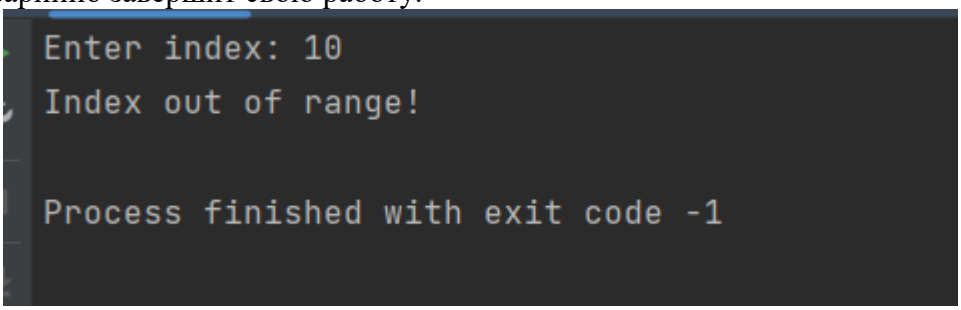

Рис. 6 – Обработка ошибок

4.2 Результаты работы задания №1

Массив для работы с функцией формируется случайным образом после выбора типа. Пункт 4 выводит этот массив.

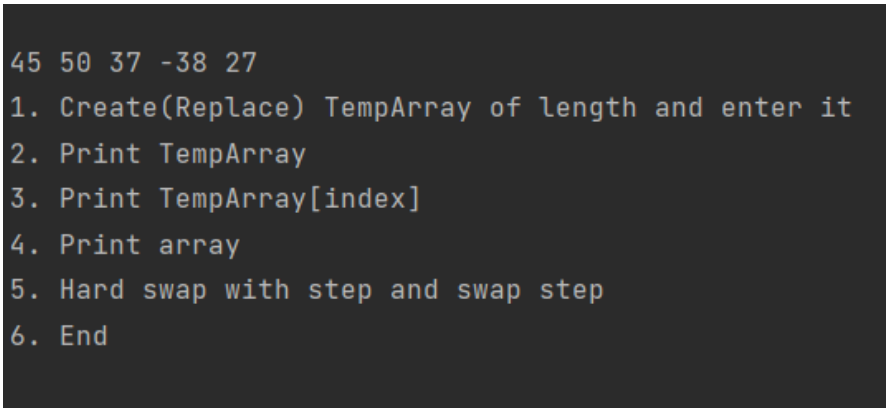

Рис. 7 – Вывод обычного массива

Пятый пункт вызывает функцию для этого массива, запрашивая шаг сдвига и шаг перестановки.

```
Enter step: 1
Enter swap: 2
1. Create(Replace) TempArray of length and enter it
2. Print TempArray
3. Print TempArray[index]
4. Print array
5. Hard swap with step and swap step
6. End
```

Рис. 8 – Запрос шагов

```
37 -38 27 50 45
1. Create(Replace) TempArray of length and enter it
2. Print TempArray
3. Print TempArray[index]
4. Print array
5. Hard swap with step and swap step
6. End
```

Рис. 9 – Вывод переставленного массива

5. Выводы

В ходе лабораторной работы был изучен метод работы с шаблонными классами и функциями. Шаблонные классы и функции выясняют свой тип во время компиляции, позволяя реализовать единые методы работы с разными типами данных. Особенностью шаблонных классов из-за природы своей работы является то, что класс реализован полностью в одном файле, линковщик не может связать шаблоны с основной программой, поскольку он работает на более позднем этапе. Также были освоены методы обработки исключений.