1 ПОСТАНОВКА ЗАДАЧИ

Создать класс для работы с одномерными массивами: перегрузить оператор ++ для увеличения элементов массива.

Память под массивы отводить динамически. Предоставить конструктор копирования. Определить friend функции для операций ввода-вывода в поток.

2 ЛИСТИНГ КОДА

Файл main.cpp

```
#include <program.h>

int main() {
    Program program;

    program.run();

    return 0;
}
```

Файл array.h

```
#pragma once

class Array {
    int *data;
    int size;

  public:
    Array();
    explicit Array(int arr_size);
    ~Array();
    Array(const Array &other);
    Array(Array &&move) noexcept;
    Array &operator=(const Array &other);
    Array &operator=(Array &&move) noexcept;
    Array &operator++();
    Array operator++(int value);

    bool isEmpty() const;
    friend void input(Array &arr, const char *msg);
    friend void show(Array arr, const char *msg);
    friend void increment(Array &arr, int num);
};
```

Файл array.cpp

```
#include "array.h"
#include "consts.h"
#include "utils.h"
#include <iostream>

Array::Array() : data(nullptr), size(0) {
}

Array::Array(int arr_size) : size(arr_size) {
    data = new int[size];
}

Array::~Array() {
    delete[] data;
    data = nullptr;
    size = 0;
}

Array::Array(const Array &other) : data(nullptr), size(other.size) {
```

```
    if (size > 0) {
        data = new int[size];

        for (int i = 0; i < size; i++) {
            data[i] = other.data[i];
        }
    }
}

Array::Array(Array &&move) noexcept : data(move.data), size(move.size) {
    move.data = nullptr;
    move.size = 0;
}

Array &Array::operator=(const Array &other) {
    if (this != &other) {
        size = other.size;

        delete[] data;
        data = nullptr;

        if (size > 0) {
            data = new int[size];

            for (int i = 0; i < size; i++) {
                data[i] = other.data[i];
            }
        }
    }

    return *this;
}

Array &Array::operator=(Array &&move) noexcept {
    if (this != &move) {
        size = move.size;

        delete[] data;
        data = move.data;

        move.data = nullptr;
        move.size = 0;
    }

    return *this;
}

Array &Array::operator++() {
    auto *new_data = new int[size + 1];

    for (int i = 0; i < size; i++) {
        new_data[i] = data[i];
    }
    new_data[size] = 0;

    delete[] data;
    data = new_data;
    new_data = nullptr;
    size++;

    return *this;
}

Array Array::operator++([[maybe_unused]] int value) {
```

```
    Array temp = *this;

    auto *new_data = new int[size + 1];

    for (int i = 0; i < size; i++) {
        new_data[i] = data[i];
    }
    new_data[size] = 0;

    delete[] data;
    data = new_data;
    new_data = nullptr;
    size++;

    return temp;
}

bool Array::isEmpty() const {
    return (data == nullptr && size == 0);
}

void input(Array &arr, const char *msg) {
    if (arr.isEmpty()) {
        while (true) {
            arr.size = getNumber("Please enter array size: ");
            if (arr.size <= 0) {
                std::cout << kRedColor << "Error, size < 0, please try
again." << kWhiteColor << std::endl;
            } else {
                break;
            }
        }
        arr.data = new int[arr.size];
    }

    std::cout << msg;

    for (int i = 0; i < arr.size; i++) {
        std ::cout << "Element " << i + 1 << ">> ";
        arr.data[i] = getNumber("");
    }
}

void show(Array arr, const char *msg) {
    std::cout << msg;

    for (int i = 0; i < arr.size; i++) {
        std::cout << arr.data[i] << " ";
    }

    std::cout << std::endl;
}

void increment(Array &arr, int num) {
    arr++;
    arr.data[arr.size - 1] = num;
}
```

Файл utils.h

```
#pragma once

int getNumber(const char *msg);
```

Файл utils.cpp

```
#include "utils.h"
#include "consts.h"
#include <iostream>

int getNumber(const char *msg) {
    int num = 0;

    std::cout << msg;

    while (true) {
        if (std::cin.peek() == '\n' || std::cin.peek() == ' ' ||
std::cin.fail()) {
            std::cin.clear();
            while (std::cin.get() != '\n' && !std::cin.eof())
                ;
            std::cout << kRedColor << "\nError, invalid input. Please try
again: " << kWhiteColor;
            continue;
        }
        if ((std::cin >> num).good() && std::cin.get() == '\n' && (kMinInt <=
num) && (num <= kMaxInt)) {
            return num;
        }
    }
}
```

Файл program.h

```
#pragma once

#include "array.h"

class Program {
    Array arr;

    void useDefaultArrayConstructor();
    void useParameterizedArrayConstructor();
    void inputArray();
    void showArray() const;
    void incrementArray();

  public:
    Program();
    void run();
};
```

Файл program.cpp

```
#include "program.h"
#include "consts.h"
#include "menus.h"
#include "utils.h"
#include <iostream>

void Program::useDefaultArrayConstructor() {
    Array tmp_arr;
    arr = tmp_arr;
```

```
    std::cout << kGreenColor << "The array object was successfully created
using the default constructor!"
              << kWhiteColor << std ::endl;
}

void Program::useParameterizedArrayConstructor() {
    int size = getNumber("Please enter the array size: ");

    Array tmp_arr(size);
    arr = tmp_arr;

    std::cout << kGreenColor << "The array object was successfully created
using the constructor with parameters!"
              << kWhiteColor << std ::endl;
}

Program::Program() {
    int opt = 0;

    system("clear");
    showConstructorsMenu();

    while (true) {
        opt = getNumber("\nPlease select a constructor menu option: ");

        switch (opt) {
        case 1:
            useDefaultArrayConstructor();
            return;
        case 2:
            useParameterizedArrayConstructor();
            return;
        default:
            std::cout << kRedColor << "\nError, you picked is an incorrect
menu option. Please try again."
                      << kWhiteColor << std::endl;
        }
    }
}

void Program::inputArray() {
    input(arr, "Please enter array elements.\n");

    std::cout << kGreenColor << "Array successfully entered using friend
function(input)!" << kWhiteColor << std::endl;
}

void Program::showArray() {
    if (arr.isEmpty()) {
        std::cout << kRedColor
                  << "\nError, array has not been entered. Please use the
first or third option and try again!"
                  << kWhiteColor << std::endl;
        return;
    }
    show(arr, "Show array on the screen: ");
    std::cout << kGreenColor << "The array was successfully displayed on the
screen using the friend function(show)!"
              << kWhiteColor << std::endl;
}

void Program::incrementArray() {
    int num = getNumber("Please enter new array element: ");
```

```
    increment(arr, num);

    std::cout << kGreenColor << "The array was successfully incremented using
the friend function(increment)!"
              << kWhiteColor << std::endl;
}

void Program::run() {
    int opt = 0;

    showTaskMenu();

    while (true) {
        opt = getNumber("\nPlease select a menu option: ");

        switch (opt) {
        case 1:
            inputArray();
            break;
        case 2:
            showArray();
            break;
        case 3:
            incrementArray();
            break;
        case 4:
            std::cout << kGreenColor << "\nYou have successfully exited the
program." << kWhiteColor << std::endl;
            return;
        default:
            std::cout << kRedColor << "\nError, you picked is an incorrect
menu option. Please try again."
                      << kWhiteColor << std::endl;
        }
    }
}
```

Файл menus.h

```
#pragma once

void showConstructorsMenu();
void showTaskMenu();
```

Файл menus.cpp

```
#include "menus.h"
#include <iostream>

void showConstructorsMenu() {
    std::cout << "\t\t\tCONSTRUCTORS MENU" << std::endl;
    std::cout << "1.Use default constructor." << std::endl;
    std::cout << "2.Use constructor with parameters." << std::endl;
}

void showTaskMenu() {
    std::cout << "\n\t\t\t\tTASK" << std::endl;
    std::cout << "Create a class for working with one-dimensional arrays." <<
std::endl;
    std::cout << "Overload the ++ operator to increment array elements." <<
std::endl;
```

```
    std::cout << "Allocate memory for arrays dynamically." << std::endl;
    std::cout << "Provide a copy constructor." << std::endl;
    std::cout << "Define friend functions for input and output operations."
<< std::endl;

    std::cout << "\n\t\t\t\tMENU" << std::endl;
    std::cout << "1. Enter array elements." << std::endl;
    std::cout << "2. Display the array." << std::endl;
    std::cout << "3. Increment array elements (++)." << std::endl;
    std::cout << "4. Exit the program." << std::endl;
}
```

Файл consts.h

```
#pragma once

inline constexpr const int kMaxInt = 2147483647;
inline constexpr const int kMinInt = -2147483648;
inline constexpr const char *kWhiteColor = "\o{33}[0m";
inline constexpr const char *kRedColor = "\o{33}[31m";
inline constexpr const char *kGreenColor = "\o{33}[32m";
```

3 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

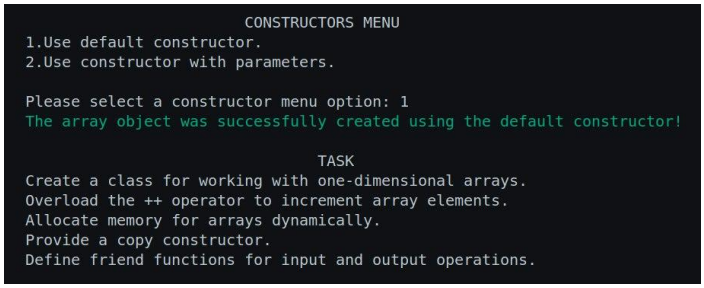

Рисунок 3.1 – Меню конструкторов класса Array и описание задания

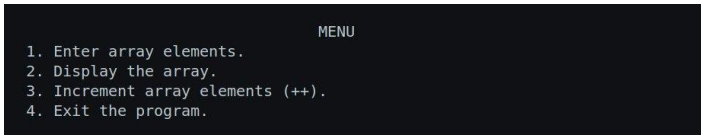

Рисунок 3.2 – Главное меню программы

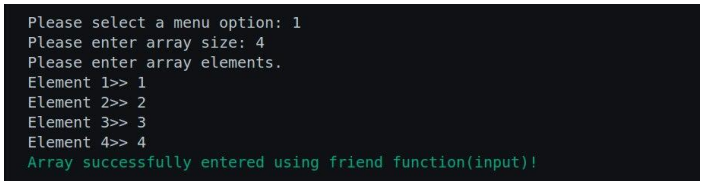

Рисунок 3.3 – Ввод элементов массива

Рисунок 3.4 – Вывод элементов массива на экран

Рисунок 3.5 – Добавление нового элемента в массив

Рисунок 3.6 – Вывод элементов массива после добавления

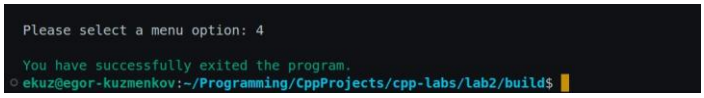

Рисунок 3.7 – Завершение работы программы

4 ЗАКЛЮЧЕНИЕ

В результате выполнения данной лабораторной работы были закреплены навыки разработки классов в C++ с использованием динамического выделения памяти. Был создан класс для работы с одномерными массивами, реализован конструктор копирования и перегружен оператор ++, позволяющий увеличивать значения элементов массива. Кроме того, были определены friend-функции для ввода и вывода данных через потоки. Проведённое тестирование подтвердило корректность реализации методов и операторов, а также показало правильность взаимодействия всех элементов программы.