main.cpp

```cpp
#include "header.h"

using namespace std;

int main() {

    Int_nums zero_arr();
    Int_nums size_arr(size_const());
    Int_nums normal_arr(arr_fill());
    Int_nums normal_arr2(arr_fill());
    Int_nums copy_arr(normal_arr);
    s_find_elem(normal_arr);
    Int_nums arr_mega_or = normal_arr || normal_arr2;
    Int_nums arr_mega_and = normal_arr && normal_arr2;
    normal_arr + normal_arr2;
    normal_arr - normal_arr2;
    normal_arr - 5;
    normal_arr + 5;

    return 0;
}
```

header.h

```cpp
#ifndef LAB2_1_HEADER_H
#define LAB2_1_HEADER_H
#include <iostream>
#include <vector>
using namespace std;

unsigned int size_const();
vector<int> arr_fill();
class Int_nums {
private:
    vector<int> nums;
public:
    Int_nums(); // EMPTY const
    Int_nums(unsigned int add_size); // MAX_SIZE const
    Int_nums(vector<int> add_nums); // NORMAL const
    Int_nums(const Int_nums &c_arr); // COPY const
    Int_nums operator+(const int &num);
    Int_nums operator-(const int &num);
    Int_nums operator&&(const Int_nums &arr_right);
    Int_nums operator||(const Int_nums &arr_right);
    Int_nums operator+(const Int_nums &arr_right);
    Int_nums operator-(const Int_nums &arr_right);
    bool c_find_elem(int &num);
    void show_array();
};
void s_find_elem(Int_nums &arr);
#endif
```

Class_procedures.cpp

```cpp
#include <iostream>
#include "header.h"
#include <algorithm>
#include <vector>
using namespace std;

// empty constructor
Int_nums::Int_nums() {
    this->nums.resize(0);
}
// SIZE constructor
Int_nums::Int_nums(unsigned int add_size) {
    this->nums.resize(add_size);
    for (unsigned int i = 0; i < add_size; i++) {
        this->nums[i] = 0;
    }
    show_array();
}
//NORMAL constructor
Int_nums::Int_nums(vector<int> add_nums){
    for (unsigned long long i = 0; i < add_nums.size(); i++) {
        this->nums.push_back(add_nums[i]);
    }
    show_array();
}
// COPY constructor
Int_nums::Int_nums(const Int_nums &c_arr) {
    for (unsigned long long i = 0; i < c_arr.nums.size(); i++) {
        this->nums.push_back(c_arr.nums[i]);
    }
    show_array();
}

Int_nums Int_nums::operator+(const int &num) {
    this->nums.push_back(num);
    return *this;
}

Int_nums Int_nums::operator-(const int &num) {
    unsigned long long size = this->nums.size(); // size RIGHT NOW
    bool check = false;
    for (unsigned long long i = 0; i < this->nums.size(); i++) {
        if (!check and this->nums[i] == num) {
            this->nums.erase(this->nums.begin() + i);
            check = true;
        }
    }
    if (size == this->nums.size()) {
        cout << "NO ELEM _" << num << "_" << endl;
    }
    return *this;
}

Int_nums Int_nums::operator&&(const Int_nums &arr_right) {
    Int_nums new_arr;
    set_intersection(this->nums.begin(), this->nums.end(), arr_right.nums.begin(),
            arr_right.nums.end(), back_inserter(new_arr.nums));
    cout << "New array (&&):" << endl;
    new_arr.show_array();
```

```
    return new_arr;
}

Int_nums Int_nums::operator||(const Int_nums &arr_right) {
    Int_nums new_arr;
    set_union(this->nums.begin(), this->nums.end(),
              arr_right.nums.begin(), arr_right.nums.end(),
              back_inserter(new_arr.nums));

    cout << "New array (||):" << endl;
    new_arr.show_array();
    return new_arr;
}

Int_nums Int_nums::operator+(const Int_nums &arr_right){
    for (unsigned long long i = 0; i < arr_right.nums.size(); i++) {
        this->nums.push_back(arr_right.nums[i]);
    }
    return *this;
}
Int_nums Int_nums::operator-(const Int_nums &arr_right) {
    for (unsigned long long i = 0; i < arr_right.nums.size(); i++) {
        int num = arr_right.nums[i];
        bool check = false;
        for (unsigned long long j = 0; j < this->nums.size(); j++) {
            if (!check and this->nums[j] == num) {
                this->nums.erase(this->nums.begin() + j);
                check = true;
            }
        }
    }
    return *this;
}

bool Int_nums::c_find_elem(int &num) {

    for (unsigned long long i = 0; i < this->nums.size(); i++) {
        if (this->nums[i] == num) return true;
    }
    return false;
}
void Int_nums::show_array() {
    cout << "SIZE:" << " " << this->nums.size() << endl;
    cout << "ELEMS:" << endl;
    for (unsigned long long i = 0; i < this->nums.size(); i++)
        cout << this->nums[i] << " ";
    cout << "\n----------------------------------------\n";
}
```

Source_procedures.cpp

```cpp
#include <iostream>
#include "header.h"
#include <vector>

using namespace std;

unsigned int size_const() {
    unsigned int max_size;
    cout << "Array size:" << endl;
    cin >> max_size;
    cout << "\n----------------------------------------\n";
    return max_size;
}

vector<int> arr_fill() {
    unsigned int max_size;
    cout << "Array size:" << endl;
    cin >> max_size;

    cout << "Array elems:" << endl;
    vector<int> arr_tmp(max_size);
    for (unsigned int i = 0; i < max_size; i++) {
        int n;
        cin >> n;
        arr_tmp[i] = n;
    }
    cout << "\n----------------------------------------\n";
    return arr_tmp;
}

void s_find_elem(Int_nums &arr) {
    cout << "What elem to find?" << endl;
    int num;
    cin >> num;
    if (!arr.c_find_elem(num)) {
        cout << "NO ELEM _" << num << "_";
    } else {
        cout << "YES ELEM _" << num << "_";
    }
}
```

Ввод	Вывод
Стандартный ввод : размер массивов и их содержимое	Стандартный вывод: размер массивов и их содержимое
Искомое значение	Конкатенация 2 строк и размер результат поиска подстроки добавление искомого элемента к строке

ЛРN№2

Микаилов Микаил Аскерович

Группа №M3110

Проверил преподаватель:
Повышев Владислав Вячеславович