МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №0**

по дисциплине: Дискретная математика

тема: «Подготовка к выполнению работы №1.1»

Выполнил: ст. группы ПВ-223

Пахомов Владислав Андреевич

Проверили:

ст. пр. Рязанов Юрий Дмитриевич

ст. пр. Бондаренко Татьяна Владимировна

Белгород 2023 г.

**Лабораторная работа № 0**

**Решения задач:**

1. Задача №1

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Получить массив С, содержащий все элементы массивов А и В без повторений.

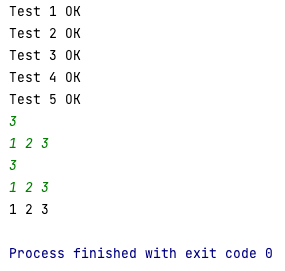
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [1, 3, 2]; B = [10, 9, 8] | C = [1, 3, 2, 10, 9, 8] |
| A = [1, 3, 2]; B = [] | C = [1, 3, 2] |
| A = []; B = [10, 9, 8] | C = [10, 9, 8] |
| A = []; B = [] | C = [] |
| A = [1, 4, 3, 2, 5]; B = [5, 1, 10] | C = [1, 4, 3, 2, 5, 10] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 2};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {10, 9, 8};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 3;  
  
 uniteArraysWithoutRepeating(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 6);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 2 &&  
 arrayC[3] == 10 &&  
 arrayC[4] == 9 &&  
 arrayC[5] == 8);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 2};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 uniteArraysWithoutRepeating(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 2);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {10, 9, 8};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 3;  
  
 uniteArraysWithoutRepeating(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 10 &&  
 arrayC[1] == 9 &&  
 arrayC[2] == 8);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 uniteArraysWithoutRepeating(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 4, 3, 2, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {5, 1, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 5, arrayBSize = 3;  
  
 uniteArraysWithoutRepeating(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 6);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 4 &&  
 arrayC[2] == 3 &&  
 arrayC[3] == 2 &&  
 arrayC[4] == 5 &&  
 arrayC[5] == 10);  
  
 printf("Test 5 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 uniteArraysWithoutRepeating(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task1.c

#include "../alg.h"  
  
void uniteArraysWithoutRepeating(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*const arrayCSize) {  
 *// Создаём указатель на первый элемент массива С* int \*cBegin = arrayC;  
  
 *// Копируем все элементы массива A в C, сдвигаем указатель на последний элемент.* for (size\_t i = 0; i < arrayASize; i++) {  
 \*(arrayC++) = arrayA[i];  
 }  
  
 *// Аналогично копируем элементы из массива B в C, кроме того проверяем,  
 // что копируемый элемент не встречается в массиве A* for (size\_t i = 0; i < arrayBSize; ++i) {  
  
 *// Проверяем, что элемента нет в массиве A* int j = 0;  
 while (j < arrayASize && arrayA[j] != arrayB[i])  
 j++;  
  
 *// Если его нет, добавляем в массив C новый элемент* if (j == arrayASize)  
 \*(arrayC++) = arrayB[i];  
 }  
  
 *// Длина итогового массива - разница указателя на последний и первый элемент* \*arrayCSize = arrayC - cBegin;  
}



1. Задача №2

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Получить массив С, содержащий все такие элементы, которые есть и в массиве А и в

массиве В.

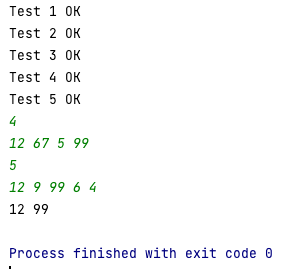
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 66, 5]; B = [17, 9, 10] | C = [] |
| A = [1, 3, 2]; B = [] | C = [] |
| A = []; B = [8, 9, 10] | C = [] |
| A = []; B = [] | C = [] |
| A = [1, 2, 3, 4, 5]; B = [5, 1, 10] | C = [5, 1] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 66, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 3;  
  
 uniteRepeatingElementsArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 2};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 uniteRepeatingElementsArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {8, 10, 9};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 3;  
  
 uniteRepeatingElementsArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 uniteRepeatingElementsArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 2, 3, 4, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {5, 1, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 5, arrayBSize = 3;  
  
 uniteRepeatingElementsArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 2);  
 **assert**(arrayC[0] == 5 &&  
 arrayC[1] == 1);  
  
 printf("Test 5 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 uniteRepeatingElementsArrays(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task2.c

#include "../alg.h"  
  
void uniteRepeatingElementsArrays(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*const arrayCSize) {  
 *// Создаём указатель на первый элемент массива С* int \*cBegin = arrayC;  
  
 *// Копируем элементы из массива B в C, кроме того проверяем,  
 // что копируемый элемент встречается в массиве A* for (size\_t i = 0; i < arrayBSize; ++i) {  
  
 *// Проверяем, что элемента есть в массиве A* int j = 0;  
 while (j < arrayASize && arrayA[j] != arrayB[i])  
 j++;  
  
 *// Если он есть, добавляем в массив новый элемент* if (j != arrayASize)  
 \*(arrayC++) = arrayB[i];  
 }  
  
 *// Длина итогового массива - разница указателя на последний и первый элемент* \*arrayCSize = arrayC - cBegin;  
}



1. Задача №3

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Получить массив С, содержащий все элементы массива А, которых нет в В.

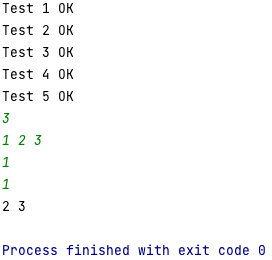
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 66, 5]; B = [17, 9, 10] | C = [4, 66, 5] |
| A = [1, 2, 3]; B = [] | C = [1, 2, 3] |
| A = []; B = [8, 9, 10] | C = [] |
| A = []; B = [] | C = [] |
| A = [1, 4, 3, 2, 5]; B = [5, 1, 10] | C = [4, 3, 2] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 100000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 5, 66};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize, arrayASize = 3, arrayBSize = 3;  
  
 getArrayWithElementsFromANotPresentInB(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 4 &&  
 arrayC[1] == 5 &&  
 arrayC[2] == 66);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 2, 3};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize, arrayASize = 3, arrayBSize = 0;  
  
 getArrayWithElementsFromANotPresentInB(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 2 &&  
 arrayC[2] == 3);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {8, 9, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize, arrayASize = 0, arrayBSize = 3;  
  
 getArrayWithElementsFromANotPresentInB(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize, arrayASize = 0, arrayBSize = 0;  
  
 getArrayWithElementsFromANotPresentInB(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 2, 3, 4, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {5, 1, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize, arrayASize = 5, arrayBSize = 3;  
  
 getArrayWithElementsFromANotPresentInB(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 2 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 4);  
  
 printf("Test 5 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 getArrayWithElementsFromANotPresentInB(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task3.c

#include "../alg.h"  
  
void getArrayWithElementsFromANotPresentInB(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*const arrayCSize) {  
 *// Создаём указатель на первый элемент массива С* int \*cBegin = arrayC;  
  
 *// Копируем элементы из массива A в C, кроме того проверяем,  
 // что копируемый элемент не встречается в массиве B* for (size\_t i = 0; i < arrayASize; ++i) {  
  
 *// Проверяем, что элемента нет в массиве B* int j = 0;  
 while (j < arrayBSize && arrayA[i] != arrayB[j])  
 j++;  
  
 *// Если его нет, добавляем в массив C новый элемент* if (j == arrayBSize)  
 \*(arrayC++) = arrayA[i];  
 }  
  
 *// Длина итогового массива - разница указателя на последний и первый элемент* \*arrayCSize = arrayC - cBegin;  
}



1. Задача №4

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Получить массив С, содержащий все элементы массива А, которых нет в В и все

элементы массива В, которых нет в А.

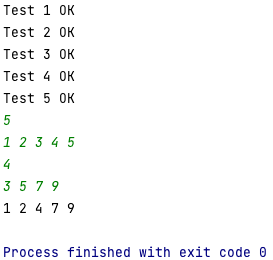
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 66, 5]; B = [17, 9, 10] | C = [4, 66, 5, 17, 9, 10] |
| A = [1, 2, 3]; B = [] | C = [1, 2, 3] |
| A = []; B = [8, 9, 10] | C = [8, 9, 10] |
| A = []; B = [] | C = [] |
| A = [1, 4, 3, 2, 5]; B = [5, 1, 10] | C = [4, 3, 2, 10] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 66, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 3;  
  
 getArrayOfNotRepeatingValues(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 6);  
 **assert**(arrayC[0] == 4 &&  
 arrayC[1] == 66 &&  
 arrayC[2] == 5 &&  
 arrayC[3] == 17 &&  
 arrayC[4] == 9 &&  
 arrayC[5] == 10);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 2, 3};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 getArrayOfNotRepeatingValues(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 2 &&  
 arrayC[2] == 3);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {8, 9, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 3;  
  
 getArrayOfNotRepeatingValues(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 8 &&  
 arrayC[1] == 9 &&  
 arrayC[2] == 10);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 getArrayOfNotRepeatingValues(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 4, 3, 2, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {5, 1, 10};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 5, arrayBSize = 3;  
  
 getArrayOfNotRepeatingValues(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 4);  
 **assert**(arrayC[0] == 4 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 2 &&  
 arrayC[3] == 10);  
  
 printf("Test 5 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 getArrayOfNotRepeatingValues(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task4.c

#include "../alg.h"  
  
void getArrayOfNotRepeatingValues(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*const arrayCSize) {  
 *// Создаём указатель на первый элемент массива С* int \*cBegin = arrayC;  
  
 *// Копируем элементы из массива A в C, кроме того проверяем,  
 // что копируемый элемент не встречается в массиве B* for (size\_t i = 0; i < arrayASize; ++i) {  
  
 *// Проверяем, что элемента нет в массиве B* int j = 0;  
 while (j < arrayBSize && arrayA[i] != arrayB[j])  
 j++;  
  
 *// Если его нет, добавляем в массив C новый элемент* if (j == arrayBSize)  
 \*(arrayC++) = arrayA[i];  
 }  
  
 *// Аналогичным образом копируем элементы из B* for (size\_t i = 0; i < arrayBSize; ++i) {  
  
 int j = 0;  
 while (j < arrayASize && arrayB[i] != arrayA[j])  
 j++;  
  
 if (j == arrayASize)  
 \*(arrayC++) = arrayB[i];  
 }  
  
 *// Длина итогового массива - разница указателя на последний и первый элемент* \*arrayCSize = arrayC - cBegin;  
}



1. Задача №5

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Определить, верно ли, что массив В содержит каждый элемент массива А.

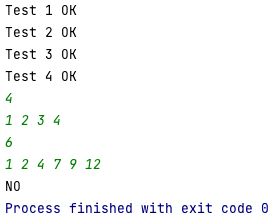
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 5, 33, 12]; B = [17, 9, 10] | NO |
| A = [17]; B = [17, 9, 10] | YES |
| A = []; B = [17, 9, 10] | YES |
| A = [17, 9, 10]; B = [17, 9, 10] | YES |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 5, 33, 12};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 4, arrayBSize = 3;  
  
 **assert**(!isBContainsEveryElementOfA(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 1, arrayBSize = 3;  
  
 **assert**(isBContainsEveryElementOfA(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 0, arrayBSize = 3;  
  
 **assert**(isBContainsEveryElementOfA(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 3, arrayBSize = 3;  
  
 **assert**(isBContainsEveryElementOfA(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 4 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 **bool** result = isBContainsEveryElementOfA(arrayA, aSize, arrayB, bSize);  
  
 if (result)  
 printf("YES");  
 else  
 printf("NO");  
  
 return 0;  
}

task5.c

#include "../alg.h"  
  
**bool** isBContainsEveryElementOfA(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize) {  
 *// Предположим, что B действительно содержит каждый элемент массива A* **bool** result = **true**;  
  
 *// Проверим, что каждый элемент A находится в B, если обнаружится что это не так, то result будет false,  
 // и смысла продолжать перебор далее не будет* for (size\_t i = 0; i < arrayASize && result; i++) {  
 *// Просто перебор* size\_t j = 0;  
 while (j < arrayBSize && arrayA[i] != arrayB[j])  
 j++;  
  
 *// Присваиваем result результат перебора, если что-то нашлось, result остаётся без изменений  
 // Иначе - становится false.* result = j != arrayBSize;  
 }  
  
 return result;  
}



1. Задача №6

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Определить, верно ли, что массивы А и В состоят из одинаковых элементов.

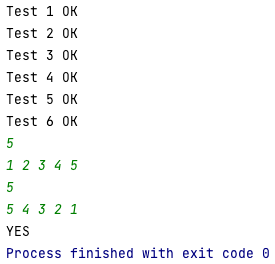
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 5, 33, 12]; B = [17, 9, 10] | NO |
| A = [10, 9, 17]; B = [17, 9, 10] | YES |
| A = []; B = [17, 9, 10] | NO |
| A = [17, 9, 10]; B = [17, 9, 10] | YES |
| A = [17, 9, 10]; B = [] | NO |
| A = []; B = [] | YES |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 5, 33, 12};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 4, arrayBSize = 3;  
  
 **assert**(!isArraysConsistsOfSameElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {10, 9, 17};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 3, arrayBSize = 3;  
  
 **assert**(isArraysConsistsOfSameElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 0, arrayBSize = 3;  
  
 **assert**(!isArraysConsistsOfSameElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 3, arrayBSize = 3;  
  
 **assert**(isArraysConsistsOfSameElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 size\_t arrayASize = 3, arrayBSize = 0;  
  
 **assert**(!isArraysConsistsOfSameElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 5 OK\n");  
}  
  
void testCase6() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 size\_t arrayASize = 0, arrayBSize = 0;  
  
 **assert**(isArraysConsistsOfSameElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 6 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
 testCase6();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 **bool** result = isArraysConsistsOfSameElements(arrayA, aSize, arrayB, bSize);  
  
 if (result)  
 printf("YES");  
 else  
 printf("NO");  
  
 return 0;  
}

task6.c

#include "../alg.h"  
  
**bool** isArraysConsistsOfSameElements(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize) {  
 *// Если массивы состоят из неповторяющихся одинаковых элементов, то логично предположить, что их  
 // размеры равны.* **bool** result = arrayASize == arrayBSize;  
  
 *// Проверим, что каждый элемент A находится в B, если обнаружится что это не так, то result будет false,  
 // и смысла продолжать перебор далее не будет* for (size\_t i = 0; i < arrayASize && result; i++) {  
 *// Просто перебор* size\_t j = 0;  
 while (j < arrayBSize && arrayA[i] != arrayB[j])  
 j++;  
  
 *// Присваиваем result результат перебора, если что-то нашлось, result остаётся без изменений  
 // Иначе - становится false.* result = j != arrayBSize;  
 }  
  
 *// Смысла проверять B нет, так как размеры массивов равны и каждому элементу A найден равный элемент из B* return result;  
}



1. Задача №7

Дано:

А – массив натуральных чисел, в котором нет одинаковых элементов;

В – массив натуральных чисел, в котором нет одинаковых элементов.

Определить, верно ли, что в массивах А и В нет общих элементов.

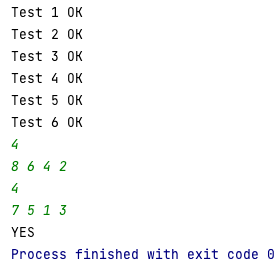
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 5, 33, 12]; B = [17, 9, 10] | YES |
| A = [10, 9, 17]; B = [17, 9, 10] | NO |
| A = []; B = [17, 9, 10] | YES |
| A = [17, 9, 10]; B = [17, 9, -10] | NO |
| A = [17, 9, 10]; B = [] | YES |
| A = []; B = [] | YES |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 5, 33, 12};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 4, arrayBSize = 3;  
  
 **assert**(isArraysConsistsOfDifferentElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {10, 9, 17};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 3, arrayBSize = 3;  
  
 **assert**(!isArraysConsistsOfDifferentElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 size\_t arrayASize = 0, arrayBSize = 3;  
  
 **assert**(isArraysConsistsOfDifferentElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {7, 9, -10};  
 size\_t arrayASize = 3, arrayBSize = 3;  
  
 **assert**(!isArraysConsistsOfDifferentElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17, 9, 10};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 size\_t arrayASize = 3, arrayBSize = 0;  
  
 **assert**(isArraysConsistsOfDifferentElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 5 OK\n");  
}  
  
void testCase6() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 size\_t arrayASize = 0, arrayBSize = 0;  
  
 **assert**(isArraysConsistsOfDifferentElements(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 6 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
 testCase6();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 **bool** result = isArraysConsistsOfDifferentElements(arrayA, aSize, arrayB, bSize);  
  
 if (result)  
 printf("YES");  
 else  
 printf("NO");  
  
 return 0;  
}

task7.c

#include "../alg.h"  
  
**bool** isArraysConsistsOfDifferentElements(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize) {  
 *// Предположим, что массивы состоят из разных элементов* **bool** result = **true**;  
  
 *// Проверим, что каждый элемент A не содержится в B, если обнаружится что это не так, то result будет false,  
 // и смысла продолжать перебор далее не будет* for (size\_t i = 0; i < arrayASize && result; i++) {  
 *// Просто перебор* size\_t j = 0;  
 while (j < arrayBSize && arrayA[i] != arrayB[j])  
 j++;  
  
 *// Присваиваем result результат перебора, если что-то нашлось, result остаётся без изменений  
 // Иначе - становится false.* result = j == arrayBSize;  
 }  
  
 return result;  
}



1. Задача №8

Даны массивы натуральных чисел А и В, упорядоченные по возрастанию.

Получить упорядоченный по возрастанию массив С, содержащий все элементы

массивов А и В.

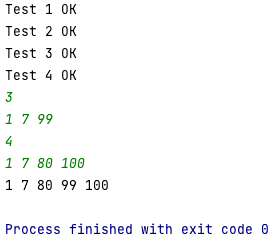
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [1, 3, 5]; B = [2, 3, 5, 6] | C = [1, 2, 3, 5, 6] |
| A = [1, 3, 5]; B = [] | C = [1, 3, 5] |
| A = []; B = [2, 3, 5, 6] | C = [2, 3, 5, 6] |
| A = []; B = [] | C = [] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 4;  
  
 uniteArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 5);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 2 &&  
 arrayC[2] == 3 &&  
 arrayC[3] == 5 &&  
 arrayC[4] == 6);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 uniteArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 5);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 4;  
  
 uniteArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 4);  
 **assert**(arrayC[0] == 2 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 5 &&  
 arrayC[3] == 6);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 uniteArrays(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 uniteArrays(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task8.c

#include "../alg.h"  
  
void uniteArrays(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*arrayCSize) {  
 *// Индексы в массиве A и B* size\_t i = 0, j = 0;  
  
 *// Пока индексы не указывают на конец массива выполняем цикл* while (i < arrayASize || j < arrayBSize) {  
 *// Первый случай. Копируем значение из A если индекс B указывает на конец массива или  
 // элемент из A меньше элемента из B* if (j == arrayBSize || (i < arrayASize && arrayA[i] < arrayB[j]))  
 arrayC[(\*arrayCSize)++] = arrayA[i++];  
 *// Второй случай. Копируем значение из B если индекс A указывает на конец массива или  
 // элемент из B меньше элемента из A* else if (i == arrayASize || arrayA[i] > arrayB[j])  
 arrayC[(\*arrayCSize)++] = arrayB[j++];  
 *// Третий случай. Элементы равны, в этом случае нужно сдвинуть любой из индексов, здесь мог быть как i, так и j* else  
 i++;  
 }  
}



1. Задача №9

Даны массивы натуральных чисел А и В, упорядоченные по возрастанию.

Получить упорядоченный по возрастанию массив С, содержащий все такие

элементы, которые есть и в массиве А и в массиве В.

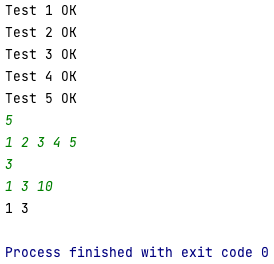
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [1, 3, 5]; B = [2, 3, 5, 6] | C = [3, 5] |
| A = [1, 3, 5]; B = [] | C = [] |
| A = []; B = [2, 3, 5, 6] | C = [] |
| A = []; B = [] | C = [] |
| A = [1, 3, 5]; B = [2, 4, 6, 8] | C = [] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 4;  
  
 uniteRepeatingElementsInSets(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 2);  
 **assert**(arrayC[0] == 3 &&  
 arrayC[1] == 5);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 uniteRepeatingElementsInSets(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 4;  
  
 uniteRepeatingElementsInSets(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 uniteRepeatingElementsInSets(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 4, 6, 8};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 4;  
  
 uniteRepeatingElementsInSets(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 5 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 uniteRepeatingElementsInSets(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task9.c

#include "../alg.h"  
  
void uniteRepeatingElementsInSets(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*arrayCSize) {  
 *// Индексы в массиве A и B* size\_t i = 0, j = 0;  
  
 *// Пока индексы не указывают на конец массива выполняем цикл* while (i < arrayASize && j < arrayBSize) {  
 *// Первый случай. A меньше элемента из B* if (i < arrayASize && arrayA[i] < arrayB[j])  
 i++;  
 *// Второй случай. A больше элемента из B* else if (j < arrayBSize && arrayA[i] > arrayB[j])  
 j++;  
 *// Третий случай. A равен элементу из B. В третьем случае копируем элемент и сдвигаем индекс массива A* else  
 arrayC[(\*arrayCSize)++] = arrayA[i++];  
 }  
}



1. Задача №10

Даны массивы натуральных чисел А и В, упорядоченные по возрастанию.

Получить упорядоченный по возрастанию массив С, содержащий все элементы

массива А, которых нет в В.

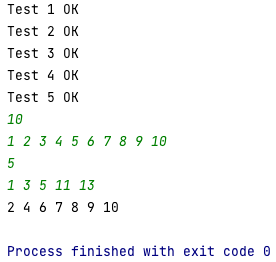
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [1, 3, 5]; B = [2, 3, 5, 6, 14] | C = [1] |
| A = [1, 3, 142]; B = [2, 3, 5, 6] | C = [1, 142] |
| A = [1, 3, 5]; B = [] | C = [1, 3, 5] |
| A = []; B = [2, 3, 5, 6] | C = [] |
| A = []; B = [] | C = [] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6, 14};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 5;  
  
 getUniqueElementsOfA(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 1);  
 **assert**(arrayC[0] == 1);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 142};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 4;  
  
 getUniqueElementsOfA(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 2);  
 **assert**(arrayC[0] == 1 && arrayC[1] == 142);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 getUniqueElementsOfA(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 && arrayC[1] == 3 && arrayC[2] == 5);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 4;  
  
 getUniqueElementsOfA(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 getUniqueElementsOfA(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 5 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 getUniqueElementsOfA(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task10.c

#include "../alg.h"  
  
void getUniqueElementsOfA(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*arrayCSize) {  
 *// Индексы в массиве A и B* size\_t i = 0, j = 0;  
  
 *// Пока индексы не указывают на конец массива выполняем цикл* while (i < arrayASize || j < arrayBSize) {  
 *// Первый случай. Элемент из A оказался меньше элемента из B или индекс j указывает на конец B  
 // Если A[i] < B[j], то в B больше никогда не встретится A[i], так как все последующие элементы  
 // будут больше B[j], поэтому можем добавлять элемент в C.  
 // Также добавляем элемент если мы достигли конца B - больше элементов не будет, и следующие элементы A  
 // в нём не встретятся* if (j == arrayBSize || (i < arrayASize && arrayA[i] < arrayB[j]))  
 arrayC[(\*arrayCSize)++] = arrayA[i++];  
 *// Второй случай. B[j] < A[i]. Здесь пока ничего не ясно, сдвигаем j индекс* else if (j < arrayBSize && arrayA[i] > arrayB[j])  
 j++;  
 *// Третий случай. Элементы равны, поэтому переходим к следующему элементу A увеличивая i* else  
 i++;  
 }  
}



1. Задача №11

Даны массивы натуральных чисел А и В, упорядоченные по возрастанию.

Получить упорядоченный по возрастанию массив С, содержащий все элементы

массива А, которых нет в В и все элементы массива В, которых нет в А.

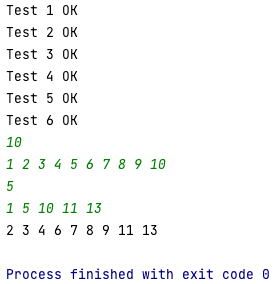
|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [1, 3, 5]; B = [2, 3, 5, 6] | C = [1, 2, 6] |
| A = [1, 3, 5]; B = [1, 3, 5] | C = [] |
| A = [1, 3, 5]; B = [2, 4, 6, 8] | C = [1, 2, 3, 4, 5, 6, 8] |
| A = [1, 3, 5]; B = [] | C = [1, 3, 5] |
| A = []; B = [2, 3, 5, 6] | C = [2, 3, 5, 6] |
| A = []; B = [] | C = [] |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 4;  
  
 uniteArraysOnlyUnique(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 2 &&  
 arrayC[2] == 6);  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 3;  
  
 uniteArraysOnlyUnique(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 4, 6, 8};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 4;  
  
 uniteArraysOnlyUnique(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 7);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 2 &&  
 arrayC[2] == 3 &&  
 arrayC[3] == 4 &&  
 arrayC[4] == 5 &&  
 arrayC[5] == 6 &&  
 arrayC[6] == 8);  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {1, 3, 5};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 3, arrayBSize = 0;  
  
 uniteArraysOnlyUnique(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 3);  
 **assert**(arrayC[0] == 1 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 5);  
  
 printf("Test 4 OK\n");  
}  
  
void testCase5() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {2, 3, 5, 6};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 4;  
  
 uniteArraysOnlyUnique(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 4);  
 **assert**(arrayC[0] == 2 &&  
 arrayC[1] == 3 &&  
 arrayC[2] == 5 &&  
 arrayC[3] == 6);  
  
 printf("Test 5 OK\n");  
}  
  
void testCase6() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
 size\_t arrayCSize = 0, arrayASize = 0, arrayBSize = 0;  
  
 uniteArraysOnlyUnique(arrayA, arrayASize, arrayB, arrayBSize, arrayC, &arrayCSize);  
  
 **assert**(arrayCSize == 0);  
  
 printf("Test 6 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
 testCase5();  
 testCase6();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize, cSize = 0;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
 int arrayC[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for (size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for (size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 uniteArraysOnlyUnique(arrayA, aSize, arrayB, bSize, arrayC, &cSize);  
  
 for (size\_t i = 0; i < cSize; i++)  
 printf("%d ", arrayC[i]);  
  
 printf("\n");  
  
 return 0;  
}

task11.c

#include "../alg.h"  
  
void uniteArraysOnlyUnique(const int \*const arrayA, const size\_t arrayASize,  
 const int \*const arrayB, const size\_t arrayBSize,  
 int \*arrayC, size\_t \*arrayCSize) {  
 size\_t i = 0, j = 0;  
  
 while (i < arrayASize || j < arrayBSize) {  
 *// Первый случай. Копируем значение из A если индекс B указывает на конец массива или  
 // элемент из A меньше элемента из B* if (j == arrayBSize || (i < arrayASize && arrayA[i] < arrayB[j]))  
 arrayC[(\*arrayCSize)++] = arrayA[i++];  
 *// Второй случай. Копируем значение из B если индекс A указывает на конец массива или  
 // элемент из B меньше элемента из A* else if (i == arrayASize || arrayA[i] > arrayB[j])  
 arrayC[(\*arrayCSize)++] = arrayB[j++];  
 *// Третий случай. Элементы равны, в этом случае нужно сдвинуть оба индекса* else {  
 j++;  
 i++;  
 }  
 }  
  
}



1. Задача №12

Даны массивы натуральных чисел А и В, упорядоченные по возрастанию.

Определить, верно ли, что массив В содержит каждый элемент массива А.

|  |  |
| --- | --- |
| Входные данные | Ожидаемый результат |
| A = [4, 5, 12, 33]; B = [9, 10, 17] | NO |
| A = [17]; B = [9, 10, 17] | YES |
| A = []; B = [9, 10, 17] | YES |
| A = []; B = [9, 10, 17] | YES |
| A = [9, 10, 17]; B = [9, 10, 17] | YES |

main.c

#include <stdio.h>  
#include <assert.h>  
  
#include "../../libs/alg/alg.h"  
  
#define **MAX\_ARRAY\_SIZE** 1000  
  
void testCase1() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {4, 5, 12, 33};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {9, 10, 17};  
 size\_t arrayASize = 5, arrayBSize = 3;  
  
 **assert**(!isBContainsEveryElementOfASets(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 1 OK\n");  
}  
  
void testCase2() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {17};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {9, 10, 17};  
 size\_t arrayASize = 1, arrayBSize = 3;  
  
 **assert**(isBContainsEveryElementOfASets(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 2 OK\n");  
}  
  
void testCase3() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {9, 10, 17};  
 size\_t arrayASize = 0, arrayBSize = 3;  
  
 **assert**(isBContainsEveryElementOfASets(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 3 OK\n");  
}  
  
void testCase4() {  
 int arrayA[**MAX\_ARRAY\_SIZE**] = {9, 10, 17};  
 int arrayB[**MAX\_ARRAY\_SIZE**] = {9, 10, 17};  
 size\_t arrayASize = 3, arrayBSize = 3;  
  
 **assert**(isBContainsEveryElementOfASets(arrayA, arrayASize, arrayB, arrayBSize));  
  
 printf("Test 4 OK\n");  
}  
  
void test() {  
 testCase1();  
 testCase2();  
 testCase3();  
 testCase4();  
}  
  
int main() {  
 test();  
  
 size\_t aSize, bSize;  
 int arrayA[**MAX\_ARRAY\_SIZE**];  
 int arrayB[**MAX\_ARRAY\_SIZE**];  
  
 scanf("%zu", &aSize);  
 for(size\_t i = 0; i < aSize; i++)  
 scanf("%d", arrayA + i);  
  
  
 scanf("%zu", &bSize);  
 for(size\_t i = 0; i < bSize; i++)  
 scanf("%d", arrayB + i);  
  
  
 **bool** result = isBContainsEveryElementOfASets(arrayA, aSize, arrayB, bSize);  
  
 if (result)  
 printf("YES");  
 else  
 printf("NO");  
  
 return 0;  
}

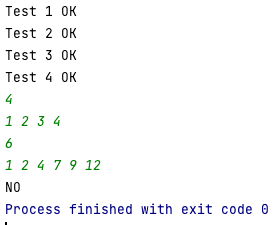
task12.c

#include "../alg.h"  
  
**bool** isBContainsEveryElementOfASets(const int \*const arrayA,

const size\_t arrayASize,

const int \*const arrayB,

const size\_t arrayBSize) {  
 size\_t i = 0, j = 0;  
 **bool** result = **true**;  
  
 while ((i < arrayASize || j < arrayBSize) && result) {  
 *// Первый случай. A[i] < B[j]. Элемент в массиве не найден, присваиваем result значение false* if (j == arrayBSize || (i < arrayASize && arrayA[i] < arrayB[j]))  
 result = **false**;  
 *// Второй случай. A[i] > B[j]. Продолжаем поиск* else if (i == arrayASize || arrayA[i] > arrayB[j])  
 j++;  
 *// Третий случай. Элементы равны, в этом случае нужно сдвинуть индекс i* else  
 i++;  
 }  
  
 return result;  
}



Статус сборки на Github: <https://github.com/IAmProgrammist/discrete_math/actions/workflows/labtests.yml>

Вывод: в ходе лабораторной работы подготовились к выполнению лабораторной работы №1.1