**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**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

автоматизированныхсистем

Дисциплина: Дискретная математика

Лабораторная работа №1

Тема: «Операции над множествами»

Выполнил:

студент группы ВТ-12

Воскобойников Илья Сергеевич

Проверил: Рязанов Ю. Д.

Белгород 2019

**Цель работы**:

изучить и научиться использовать алгебру подмножеств, изучить различные способы представления множеств в памяти ЭВМ, научиться программно реализовывать операции над множествами и выражения в алгебре подмножеств.

Вариант 4

**Задание 1**

1.Вычислить значения выражения в соответствии с вариантном. Во всех вариантах считать U={1,2,3,4,5,6,7,8,9,10}. Решение изобразить с помощью кругов Эйлера.

|  |  |  |
| --- | --- | --- |
| D=(A - (BΔC)) ∪ ((B∪C)-A) | | |
| A={1,2,3,4,6,7} | B={1,3,6,7} | C={3,4,5,6,8} |

**Выполнение**

|  |  |
| --- | --- |
| **Описание решения** | **Изображение решения** |
| BΔC:  {1,3,6,7}Δ{3,4,5,6,8}={1,4,5,7,8} |  |
| A - (BΔC):  {1,2,3,4,6,7}-{1,4,5,7,8}={2,3,6} |  |
| B∪C:  {1,3,6,7}∪{3,4,5,6,8}={1,3,4,5,6,7,8} |  |
| (B∪C)-A:  {1,3,4,5,6,7,8}-{1,2,3,4,6,7}={5,8} |  |
| (A - (BΔC)) ∪ ((B∪C)-A):  {2,3,6}∪{5,8}={2,3,5,6,8} |  |

**Итог:**

D=({1,2,3,4,6,7}-({1,3,6,7}Δ{3,4,5,6,8}))∪(({1,3,6,7}∪{3,4,5,6,8})-{1,2,3,4,6,7})={2,3,5,6,8}

**Задание 2**

Записать выражения в алгебре подмножеств, значение которого при заданных множествах А, В и С равно множеству D.

|  |  |  |  |
| --- | --- | --- | --- |
| A={1,2,3,8} | B={3,6,7} | C={2,3,4,5,7} | D={1,3,6,8} |

**Выполнение**

**1 способ:**

1. A∪B={1,2,3,8}∪{3,6,7}={1,2,3,6,7,8}

Полученное в результате объединения множество содержит все элементы множества D, но нужно избавится от элементов 2 и 7.

1. A∪B-С={1,2,3,6,7,8}-{2,3,4,5,7}={1,6,8}

Путем вычитания из A∪B множества С, мы избавились от элементов 2 и 7, но при этом потеряли элемент 3.

1. A∩B∩C={1,2,3,8}∩{3,6,7}∩{2,3,4,5,7}={3}

В результате пересечения множеств А,В и С, мы получили не достающий элемент 3.

1. (A∪B-С) ∪ A∩B∩C={1,6,8}∪{3}={1,3,6,8}

При объединении (A∪B-С) и A∩B∩C, мы получили множество D

Таким образом,

**D=(A∪B-С) и A∩B∩C**

**2 способ:**

1. В∪С={3,6,7}∪{2,3,4,5,7}={2,3,4,5,6,7}

Полученное в результате объединения множество содержит часть элементы множества D, но нужно избавится от элементов 2,4,5 и 7 и добавить элементы 1 и 8.

1. A - В∪С ={1,2,3,6,7,8}-{2,3,4,5,6,7}={1,8}

Путем вычитания из множества A множества В∪С, мы избавились от элементов 2,4,5 и 7, и добавили элементы 1 и 8, но при этом потеряли элементы 3 и 6.

1. A∩B∩C={1,2,3,8}∩{3,6,7}∩{2,3,4,5,7}={3}

В результате пересечения множеств А,В и С, мы получили не достающий элемент 3.

1. (A - В∪С) ∪ A∩B∩C={1,8}}∪{3}={1,3,8}

При объединении (A - В∪С) и A∩B∩C, у нас не хватает только одного элемента.

1. A∪С={1,2,3,6,7,8}∪{2,3,4,5,7}={1,2,3,4,5,6,7,8}

При объединении множество А и множества С, мы получили множество из всех элементов кроме элемента 6.

1. В-A∪С={3,6,7}-{1,2,3,4,5,6,7,8}={6}

В результате вычитания из множества В множества A∪С, мы получили наш не достающий элемент 6

1. ((A - В∪С) ∪ A∩B∩C) ∪( В-A∪С)= {1,3,8}∪{6}={1,3,6,8}

При объединении ((A - В∪С) ∪ A∩B∩C) и ( В-A∪С) мы получили множество D.

Таким образом,

**D=((A - В∪С) ∪ A∩B∩C) ∪( В-A∪С)**

**Задание 3**

Программно реализовать операции над множествами, используя следующие способы представления множества в памяти ЭВМ:

а) элементы множества А хранятся в массиве А. Элементы массива А неупорядочены;

б) элементы множества А хранятся в массиве А. Элементы массива А упорядочены по возрастанию;

в) элементы множества А хранятся в массиве А, элементы которого типа boolean. Если i ∈ A, то Аi = true, иначе Ai = false.

**Выполнение**

Операции над множествами

1. Включение А в В (А ⊆ В или В ⊇ А) истинно, если каждый элемент множества А принадлежит множеству В.

2.Равенство А и В (А=В). Множества А и В равны, если А ⊆ В и В ⊇ А

3. Строгое включение А в В (А ⊂ В или В ⊃ А) истинно, если А ⊆ В и А ≠ В. Если А ≠ ∅ и А ⊂ В , то А есть собственное подмножество В.

4. Объединение А и В (А ∪ В) есть множество, состоящее из всех тех и только тех элементов, которые принадлежат А или В, т.е. А ∪ В = {x | x ∈ A или x ∈ B}.

5. Пересечение А и В (А ∩ В) есть множество, состоящее из всех тех и только тех элементов, которые принадлежат каждому из множеств А и В, т.е. А ∩ В = {x | x ∈ A и x ∈ B}.

6. Разность А и В (А – В) есть множество, состоящее из всех тех и только тех элементов множества А, которые не принадлежат множеству В, т.е. А – В = {x | x ∈ A и x ∉ B}. Для обозначения разности множеств в дискретной математике обычно используют символ ―\‖.

7. Симметрическая разность А и В (А Δ В) есть множество, состоящее из всех тех и только тех элементов множества А, которые не принадлежат множеству В и только тех элементов множества В, которые не принадлежат множеству А, т.е. А Δ В = {x | x ∈ A и x ∉ B или x ∈ В и x ∉ А}. А Δ В = (А ∪ В) – (А ∩ В) = (А – В) ∪ (В – А).

8. Дополнение А до универсума U ( A ) есть множество, состоящее из всех тех и только тех элементов универсума U, которые не принадлежат множеству А, т.е. A = {x | x ∉ А}. A = U – A.

**Описание спецификаций подпрограмм**

Подпрограммы мы для выполнения операций 1-3: (а,б)

♦ Включение

Функция A\_vkl\_B

**1**.Заголовок: \_Bool A\_vkl\_B (int\* a, int\* b, int n, int m )

**2**.Назначение: возвращает true , если множество А включено в множество В, иначе -false;

**3**.Входные параметры: а, b, n, m;

**4.**Выходные параметры: нет.

♦ Равенство

Функция A\_ravn\_B

**1.**Заголовок: \_Bool A\_ravn\_B (int\* a, int\* b, int n, int m )

**2.**Назначение: возвращает true, если множества А и В равны и false в противном случае;

**3**.Входные параметры: а, b, n, m;

**4.**Выходные параметры: нет.

♦ Строгое включение

Функция A\_st\_vkl\_B

**1**.Заголовок: \_Bool A\_st\_vkl\_B (int\* a, int\* b, int n, int m )

**2**.Назначение: возвращает true, если множество А строго включено в множество В, иначе — *false;*

**3.**Входные параметры: а, b, n, m;

**4**.Выходные параметры: нет.

(в)

♦ Включение

Функция A\_vkl\_B

**1**.Заголовок: \_Bool A\_vkl\_B (int\* a, int\* b, int n )

**2**.Назначение: возвращает true , если множество А включено в множество В, иначе -false;

**3**.Входные параметры: а, b, n;

**4.**Выходные параметры: нет.

♦ Равенство

Функция A\_ravn\_B

**1.**Заголовок: \_Bool A\_ravn\_B (int\* a, int\* b, int n )

**2.**Назначение: возвращает true, если множества А и В равны и false в противном случае;

**3**.Входные параметры: а, b, n;

**4.**Выходные параметры: нет.

♦ Строгое включение

Функция A\_st\_vkl\_B

**1**.Заголовок: \_Bool A\_st\_vkl\_B (int\* a, int\* b, int n )

**2**.Назначение: возвращает true, если множество А строго включено в множество В, иначе — *false;*

**3.**Входные параметры: а, b, n;

**4**.Выходные параметры: нет.

*Подпрограммы для* выполнения операций 4-8

(случаи а - б)

*♦*Объединение

Функция А\_ог\_В

**1***.* Заголовок*:* int A\_or\_B (int\* a, int\* b, int n, int m, int\* c)

**2**. Назначение: в переменную *с* записывает результат объединения множеств А и B

и возвращает длину массива *с* (мощность множества С);

**3.** Входные параметры: a, b, n, m;

**4**. Выходные параметры: с.

*♦*Пересечение

Функция А\_and\_В

**1**. Заголовок: int A\_and\_B (int\* a, int\* b, int n, int m,int\* c )

**2.** Назначение: в переменную c записывает результат пересечения множеств А и В и возвращает длину массива с (мощность множества C);

**3**. Входные параметры: а, b, n, m;

**4**.Выходные параметры: с.

*♦*Разность

Функция A\_minus\_B

**1**. Заголовок int A\_minus\_B (int\* a, int\* b, int n, int m,int\* c )

**2**. Назначение: в переменную с записывает результат разности множеств А и Ви возвращает длину массива c (мощность полученного множества);

**3**. Входные параметры: а, b, n, m;

**4**.Выходные параметры: c.

*♦*Симметрическая разность

Функция A\_sm\_minus\_B

**1.**Заголовок: int A\_sm\_minus\_B (int\* a, int\* b, int n, int m,int\* c )

**2**. Назначение: в переменную c записывает результат симметрической разности множеств A и В и возвращает длину массива c (мощность полученного множества);

**3**. Входные параметры: а, b, n, m;

**4**. Выходные параметры: c.

*♦*Дополнение

Функция dop\_A

**1**. Заголовок: int dop\_A (int\* a, int n, int\* c ))

**2.** Назначение: в переменную с записывает результат дополнения множества А до универсума и возвращает длину массива с (мощность полученного множества);

**3**. Входные параметры: а, b, n, m;

**4**. Выходные параметры: c.

Подпрограммы для выполнения операций 4-8 (случай в)

*♦*Объединение

Функция A\_or\_B

**1.** Заголовок: void A\_or\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c)

**2.** Назначение: в переменную с записывает результат объединения множеств А и B;

**3**. Выходные параметры: a, b;

**4**. Выходные параметры: с.

*♦*Пересечение

Процедура A\_and\_B

**1**. Заголовок: void A\_and\_B (\_Bool a[], \_Bool b[], \_Bool c[])

**2**. Назначение: в с записывает результат пересечения множеств А и В;

**3.** Входные параметры: а, b;

**4**. Выходные параметры: c.

*♦*Разность

Процедура A\_minus\_B

**1**. Заголовок: void A\_minus\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c)

**2**. Назначение: в переменную с записывает результат разности множеств А и В;

**3**. Входные параметры: а, b;

**4**. Выходные параметры: с.

*♦*Симметрическая разность

Процедура A\_sm\_minus\_B

**1**. Заголовок void A\_sm\_minus\_B (\_Bool\* a, \_Bool\* b, int n, int m , \_Bool\* c )

**2**. Назначение: в переменную с записывает результат симметрической разности множеств А и В ;

**3**. Входные параметры: а, b;

**4.** Выходные параметры: c.

*♦*Дополнение

Процедура dop\_A

**1**. Заголовок: int dop\_A (\_Bool \* a, int n, \_Bool\* c )

**2**. Назначение: в переменную с записывает результат дополнения множества А до универсума;

**3**. Входные параметры: а;

**4**. Выходные параметры: с.

**Задание 4**

**А)**

#ifndef LAB\_1\_ZAGALOVOK\_H\_ARR\_NO\_SORT\_H

#define LAB\_1\_ZAGALOVOK\_H\_ARR\_NO\_SORT\_H

void del\_el(int\* a, int \*n);

void input (int a[], int\* n);

void output (int a[],int n);

\_Bool poisk\_el (int a[],int N,int key);

\_Bool A\_vkl\_B (int\* a, int\* b, int n, int m);

\_Bool A\_ravn\_B (int\* a, int\* b, int n, int m);

\_Bool A\_st\_vkl\_B (int\* a, int\* b, int n, int m);

int A\_or\_B (int\* a, int\* b, int n, int m,int\* c );

int A\_and\_B (int\* a, int\* b, int n, int m,int\* c );

int A\_minus\_B (int\* a, int\* b, int n, int m,int\* c );

int A\_sm\_minus\_B (int\* a, int\* b, int n, int m,int\* c );

int dop\_A (int\* a, int n, int\* c );

#endif //LAB\_1\_ZAGALOVOK\_H\_ARR\_NO\_SORT\_H

#include <stdio.h>

#include <stdbool.h>

void del\_el(int\* a, int \*n){

int j,i =0;

while (i<\*n){

j=i+1;

while (j<\*n){

if (a[j]==a[i]){

for (int k = j; k <\*n ; ++k) {

a[k]=a[k+1];

}

--\*n;

}

else{

j++;

}

}

i++;

}

}

void input (int\* a,int\* n){

for(int i=0;i<\*n;i++) {

scanf("%d", &a[i]);

}

del\_el(a,&\*n);

}

void output (int\* a,int n) {

for (int i = 0; i < n; i++)

printf("%d ",a[i]);

printf("\n");

}

\_Bool poisk\_el (int\* a,int n,int key){

int i=0;

while ((i<n) && (key!=a[i])) {

i++;

}

return i<n;

}

\_Bool A\_vkl\_B (int\* a, int\* b, int n, int m ){

\_Bool F=(n<=m);

int i=0;

while (i<n && F==true){

F=poisk\_el(b,m,a[i]);

i++;

}

return F;

}

\_Bool A\_ravn\_B (int\* a, int\* b, int n, int m ){

\_Bool F= (n==m);

int i=0;

while (i<n && F==true){

F=poisk\_el(b,m,a[i]);

i++;

}

return F;

}

\_Bool A\_st\_vkl\_B (int\* a, int\* b, int n, int m ){

\_Bool F=false;

if(n<m){

F=A\_vkl\_B(a,b,n,m);

return F;

}

int A\_or\_B (int\* a, int\* b, int n, int m, int\* c){

for (int i= 0; i < n; ++i) {

c[i]=a[i];

}

int kc=n;

for (int i = 0; i < m; ++i) {

if (!poisk\_el(c,kc,b[i])){

c[kc++]=b[i];

}

}

return kc;

}

int A\_and\_B (int\* a, int\* b, int n, int m,int\* c ){

int kc=0;

for (int i = 0; i < n ; ++i) {

if(poisk\_el(b,m,a[i])){

c[kc++]=a[i];

}

}

return kc;

}

int A\_minus\_B (int\* a, int\* b, int n, int m,int\* c ){

int kc=0;

for (int i = 0; i <n ; ++i) {

if(!poisk\_el(b,m,a[i])) {

c[kc++] = a[i];

}

}

return kc;

}

int A\_sm\_minus\_B (int\* a, int\* b, int n, int m,int\* c ){

int kc=0;

for (int i = 0; i <n; ++i) {

if(!poisk\_el(b,m,a[i])){

c[kc++]=a[i];

}

}

for (int i = 0; i <m; ++i) {

if(!poisk\_el(a,n,b[i])){

c[kc++]=b[i];

}

}

return kc;

}

int dop\_A (int\* a, int n, int\* c ){

int kc=0;

for (int u = 1; u <= 10 ; ++u) {

if(!poisk\_el(a,n,u)){

c[kc++]=u;

}

}

return kc;

}

**Б)**

#ifndef LAB\_1\_3\_A\_ZAGALOVOK\_ARR\_SORT\_H

#define LAB\_1\_3\_A\_ZAGALOVOK\_ARR\_SORT\_H

\_Bool check\_sort(int\* a, int n);

void del\_el(int\* a, int \*n);

void input (int a[], int\* n);

void output (int a[],int n);

\_Bool A\_vkl\_B (int\* a, int\* b, int n, int m);

\_Bool A\_ravn\_B (int\* a, int\* b, int n, int m);

\_Bool A\_st\_vkl\_B (int\* a, int\* b, int n, int m);

int A\_or\_B (int\* a, int\* b, int n, int m,int\* c );

int A\_and\_B (int\* a, int\* b, int n, int m,int\* c );

int A\_minus\_B (int\* a, int\* b, int n, int m,int\* c );

int A\_sm\_minus\_B (int\* a, int\* b, int n, int m,int\* c );

int dop\_A (int\* a, int n, int\* c );

void sort\_vst(int\* a, int n);

#endif //LAB\_1\_3\_A\_ZAGALOVOK\_ARR\_SORT\_H

#include <stdio.h>

#include <stdbool.h>

void del\_el(int\* a, int \*n){

int j,i =0;

while (i<\*n){

j=i+1;

while (j<\*n){

if (a[j]==a[i]){

for (int k = j; k <\*n ; ++k) {

a[k]=a[k+1];

}

--\*n;

}

else{

j++;

}

}

i++;

}

}

\_Bool check\_sort(int\* a, int n){

int i=0;

while ((i<n-1)&&(a[i]<a[i+1])){

i++;

}

return i==n;

}

void sort\_vst(int\* a, int n){

int x,j;

for (int i = 0; i < n ; ++i) {

x=a[i];

j=i;

while ((j>0)&&(a[j-1]>x)){

a[j]=a[j-1];

j--;

}

a[j]=x;

}

}

void input (int\* a,int\* n){

for(int i=0;i<\*n;i++) {

scanf("%d", &a[i]);

}

del\_el(a,&\*n);

if (!check\_sort(a,\*n)){

sort\_vst(a,\*n);

}

}

void output (int\* a,int n) {

for (int i = 0; i < n; i++)

printf("%d ",a[i]);

printf("\n");

}

\_Bool A\_vkl\_B (int\* a, int\* b, int n, int m ){

\_Bool F=((n<=m)&&(a[n]<=b[m]));

int i=0,j=0;

while ((i<n) && (F==true)) {

if (a[i] == b[j]) {

i++;

j++;

}

else {

if (a[i] > b[j]) {

j++;

}

else {

F = false;

}

}

}

return F;

}

\_Bool A\_ravn\_B (int\* a, int\* b, int n, int m ){

\_Bool F= (n==m);

int i=0;

while (i<n && F==true){

F=(a[i]==b[i]);

i++;

}

return F;

}

\_Bool A\_st\_vkl\_B (int\* a, int\* b, int n, int m ){

\_Bool F=(n < m) && A\_vkl\_B(a,b,n,m);

return F;

}

int A\_or\_B (int\* a, int\* b, int n, int m, int\* c){

int i=0,j=0,kc=0;

while ((i<n)&&(j<m)){

if (a[i]==b[j]){

c[kc]=a[i];

i++;

j++;

}

else{

if(a[i]>b[j]){

c[kc]=b[j];

j++;

}

else{

c[kc]=a[i];

i++;

}

}

kc++;

}

while (i<n){

c[kc++]=a[i];

i++;

}

while (j<m){

c[kc++]=b[j];

j++;

}

return kc;

}

int A\_and\_B (int\* a, int\* b, int n, int m,int\* c ){

int i=0,j=0,kc=0;

while ((i<n)&&(j<m)){

if (a[i]==b[j]){

c[kc++]=a[i];

i++;

j++;

}

else {

if(a[i]>b[j]){

j++;

}

else{

i++;

}

}

}

return kc;

}

int A\_minus\_B (int\* a, int\* b, int n, int m,int\* c ){

int i=0,j=0,kc=0;

while ((i<n)&&(j<m)){

if(a[i]==b[j]){

i++;

j++;

}

else{

if (a[i]>b[j]){

j++;

}

else{

c[kc++]=a[i];

i++;

}

}

}

while (i<n){

c[kc++]=a[i];

i++;

}

return kc;

}

int A\_sm\_minus\_B (int\* a, int\* b, int n, int m,int\* c ){

int i=0,j=0,kc=0;

while ((i<n)&&(j<m)){

if (a[i]==b[j]){

i++;

j++;

}

else{

if(a[i]>b[j]){

c[kc++]=b[j];

j++;

}

else{

c[kc++]=a[i];

i++;

}

}

}

while (i<n){

c[kc++]=a[i];

i++;

}

while (j<m){

c[kc++]=b[j];

j++;

}

return kc;

}

int dop\_A (int\* a, int n, int\* c ){

int u=1, j=0,kc=0;

while ((u<=10)&&(j<n)){

if (u==a[j]){

u++;

j++;

}

else{

if (u>a[j]){

j++;

}

else{

c[kc++]=u;

u++;

}

}

}

while (u<=10){

c[kc++]=u;

u++;

}

}

**В)**

#ifndef LAB\_1\_3\_A\_ZAGALOVOK\_BOOL\_ARR\_H

#define LAB\_1\_3\_A\_ZAGALOVOK\_BOOL\_ARR\_H

void input (\_Bool\* a, int\* n);

void output (\_Bool \* a,int n);

\_Bool A\_vkl\_B (\_Bool \* a, \_Bool \* b, int n );

\_Bool A\_ravn\_B (\_Bool \* a, \_Bool \* b, int n );

\_Bool A\_st\_vkl\_B (\_Bool \* a, \_Bool \* b, int n, int m );

void A\_or\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c);

void A\_and\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c );

void A\_minus\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c);

void A\_sm\_minus\_B (\_Bool\* a, \_Bool\* b, int n, int m , \_Bool\* c );

int dop\_A (\_Bool \* a, int n, \_Bool\* c );

#endif //LAB\_1\_3\_A\_ZAGALOVOK\_BOOL\_ARR\_H

#include <stdio.h>

#include <stdbool.h>

void input (\_Bool\* a, int\* n){

int j;

for (int i = 0; i <n ; ++i) {

a[i]=false;

}

for (int i = 0; i <n ; ++i) {

printf("%d",j);

a[j-1]=true;

}

}

void output (\_Bool \* a,int n) {

for (int i = 0; i < n; ++i) {

if (a[i]){

printf("%d",i+1);

}

}

}

\_Bool A\_vkl\_B (\_Bool \* a, \_Bool \* b, int n ){

int i=0;

\_Bool F=true;

while ((i<n)&&(F==true)){

F=(a[i]<=b[i]);

i++;

}

return F;

}

\_Bool A\_ravn\_B (\_Bool \* a, \_Bool \* b, int n ){

int i=0;

\_Bool F=true;

while ((i<n)&&(F==true)){

F=(a[i]==b[i]);

i++;

}

return F;

}

\_Bool A\_st\_vkl\_B (\_Bool \* a, \_Bool \* b, int n, int m ){

\_Bool F=(n < m) && A\_vkl\_B(a,b,n);

return F;

}

void A\_or\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c){

for (int i = 0; i < n; ++i) {

c[i]=(a[i]||b[i]);

}

}

void A\_and\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c ){

for (int i = 0; i < n; ++i) {

c[i]=(a[i]&&b[i]);

}

}

void A\_minus\_B (\_Bool\* a, \_Bool\* b, int n, \_Bool\* c){

for (int i = 0; i < n; ++i) {

c[i]=(a[i]>b[i]);

}

}

void A\_sm\_minus\_B (\_Bool\* a, \_Bool\* b, int n, int m , \_Bool\* c ){

for (int i = 0; i < n; ++i) {

c[i]=(a[i]!=b[i]);

}

}

int dop\_A (\_Bool \* a, int n, \_Bool\* c ){

for (int i = 0; i < n; ++i) {

c[i]=!a[i];

}

}

**Задание 5**

#include <stdio.h>

#include <process.h>

//#include "zagalovok\_arr\_no\_sort.h" //библиотека для хранения и обработки множества как

//не отсортированного массива натуральных чисел

#include "zagalovok\_arr\_sort.h" //библиотека для хранения и обработки множества как

//сортированного массива натуральных чисел

//#include "zagalovok\_bool\_arr.h" //библиотека для хранения и обработки множества как массива

//с баззовым типом boolean

int main() {

int na,nb,nc,nd,nm1,nm2;

int a[100], b[100], c[100], d[100], m1[100], m2[100];

printf("Униврсум множеств, доступных для ввода: {1,2,3,4,5,6,7,8,9,10} \n");

printf("Количество элементов множества >0 И <=10 \n");

printf("Введите количество элементов A: ");

scanf("%d",&na);

printf("Введите множество A: ");

input(a,&na);

printf("Введите количество элементов B: ");

scanf("%d",&nb);

printf("Введите множество B: ");

input(b,&nb);

printf("Введите количество элементов C: ");

scanf("%d",&nc);

printf("Введите множество C: ");

input(c,&nc);

nd=A\_sm\_minus\_B(b,c,nb,nc,d);

nm1=A\_minus\_B(a,d,na,nd,m1);

nd=A\_or\_B(b,c,nb,nc,d);

nm2=A\_minus\_B(d,a,nd,na,m2);

nd=A\_or\_B(m1,m2,nm1,nm2,d);

printf("Результат:");

output(d,nd);

system("pause");

return 0;

}