|  |
| --- |
| Федеральное государственное автономное  образовательное учреждение  высшего профессионального образования  «СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  институт  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  кафедра |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  тема  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Преподаватель \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата инициалы, фамилия  Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_  номер группы, зачетной книжки подпись, дата инициалы, фамилия |

# Задание

Написать программу, на языке С++ в среде Visual Studio для определения, является ли множество С объединением множеств A и B (AB), пересечением множеств (AB), разностью множеств А и B (A\B), разностью множеств В и А (В\А)

# Операции со множествами

Элементы множества задаются целочисленными идентификаторами

Составляем полный список всех используемых идентификаторов элементов множеств

Сопоставляем каждому множеству двоичный вектор

Двоичный вектор флагов объединения множеств A и B x1=a or b

Двоичный вектор флагов пересечения множеств A и B x2=a and b

Двоичный вектор флагов разности множеств А и B x3=a sub b

Двоичный вектор флагов разности множеств B и A x4=b sub a

|  |  |  |
| --- | --- | --- |
| **or** | **0** | **1** |
| **0** | 0 | 1 |
| **1** | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| **and** | **0** | **1** |
| **0** | 0 | 0 |
| **1** | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| **sub** | **0** | **1** |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| **xor** | **0** | **1** |
| **0** | 0 | 1 |
| **1** | 1 | 0 |

Двоичный вектор x==y тогда и только тогда, когда count(x xor y)==0 где count – количество ненулевых элементов в векторе

Программа работает в 2-х режимах:

* с выводом результатов работы на экран (для демонстрации работоспособности);
* без вывода результатов работы на экран, но с определением времени, затрачиваемого на вычисления;

Листинг кода

#include "Header.hpp"

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Обнуление двоичного вектора

vector - указатель на массив

count - количество элементов

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void clear(int \*vector, int count)

{

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

vector[i]=0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Число ненулевых элементов двоичного вектора

vector - указатель на массив

count - количество элементов

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int sum(int \*vector, int count)

{

int s=0;

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

s+=vector[i];

return s;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Булева операция AND элементов двоичных векторов

vector1 - указатель на массив

vector2 - указатель на массив

result - указатель на массив

count - количество элементов

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void and(int \*vector1, int \*vector2,int \*result, int count)

{

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

result[i]=vector1[i]&vector2[i];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Булева операция OR элементов двоичных векторов

vector1 - указатель на массив

vector2 - указатель на массив

result - указатель на массив

count - количество элементов

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void or(int \*vector1, int \*vector2,int \*result, int count)

{

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

result[i]=vector1[i]|vector2[i];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Булева операция XOR элементов двоичных векторов

vector1 - указатель на массив

vector2 - указатель на массив

result - указатель на массив

count - количество элементов

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void xor(int \*vector1, int \*vector2,int \*result, int count)

{

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

result[i]=vector1[i]^vector2[i];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Булева операция SUB элементов двоичных векторов

vector1 - указатель на массив

vector2 - указатель на массив

result - указатель на массив

count - количество элементов

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sub(int \*vector1, int \*vector2,int \*result, int count)

{

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

result[i]=vector1[i]&(vector1[i]^vector2[i]);

}

#include "Header.hpp"

using namespace std;

// 14. Определить, является ли множество С

// объединением множеств A и B (A B),

// пересечением множеств A и B (A B),

// разностью множеств А и B (A\B),

// разностью множеств В и А (В\А).

// элементы множества задаются целочисленными идентификаторами

int main()

{

int \*a; // идентификаторы элементов множества A

int \*b; // идентификаторы элементов множества B

int \*c; // идентификаторы элементов множества C

int na; // количество элементов множества A

int nb; // количество элементов множества B

int nc; // количество элементов множества C

int demo; // режим программы

setlocale(LC\_ALL, "Russian");

cout << "\tОпределить, является ли множество С" << endl;

cout << "\tобъединением множеств A и B" << endl;

cout << "\tпересечением множеств A и B" << endl;

cout << "\tразностью множеств А и B" << endl;

cout << "\tразностью множеств В и А" << endl;

cout << "\tВведите количество элементов множества A:" << endl; cin >> na;

a = new int[na+1]; // +1 для случая пустого множества

cout << "\tВведите идентификаторы элементов множества A["<<na<<"]:" << endl;

for(int i=0;i<na;i++) cin>>a[i];

cout << "\tВведите количество элементов множества B:" << endl; cin >> nb;

b = new int[nb+1]; // +1 для случая пустого множества

cout << "\tВведите идентификаторы элементов множества B["<<nb<<"]:" << endl;

for(int i=0;i<nb;i++) cin>>b[i];

cout << "\tВведите количество элементов множества C:" << endl; cin >> nc;

c = new int[nc+1]; // +1 для случая пустого множества

cout << "\tВведите идентификаторы элементов множества C["<<nc<<"]:" << endl;

for(int i=0;i<nc;i++) cin>>c[i];

cout << "\tДемонстрационный режим програмы (0 - замер времени, 1 - демонстрация работоспособности):" << endl; cin >> demo;

clock\_t t = clock();

// Составляем полный список всех используемых идентификаторов элементов множеств

int \*ids = new int[na+nb+nc+1]; // Список всех используемых идентификаторов +1 для случая пустого множества

int count=0; // Количество используемых идентификаторов

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

int id = a[i];

int index=count;

while(index-->0) if(ids[index]==id) break;

if(index==-1) ids[count++]=id;

}

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

int id = b[i];

int index=count;

while(index-->0) if(ids[index]==id) break;

if(index==-1) ids[count++]=id;

}

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

int id = c[i];

int index=count;

while(index-->0) if(ids[index]==id) break;

if(index==-1) ids[count++]=id;

}

// Сопоставляем каждому множеству двоичный вектор

int \*va = new int[count+1]; // Двоичный вектор флагов множества A

int \*vb = new int[count+1]; // Двоичный вектор флагов множества B

int \*vc = new int[count+1]; // Двоичный вектор флагов множества C

clear(va, count);

clear(vb, count);

clear(vc, count);

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

int id = a[i];

int index=count;

while(index-->0) if(ids[index]==id) break;

va[index]=1;

}

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

int id = b[i];

int index=count;

while(index-->0) if(ids[index]==id) break;

vb[index]=1;

}

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

int id = c[i];

int index=count;

while(index-->0) if(ids[index]==id) break;

vc[index]=1;

}

if(demo!=0){

cout << "\tВектор a:"; for(int i=0;i<count;i++) cout<<va[i]; cout << endl;

cout << "\tВектор b:"; for(int i=0;i<count;i++) cout<<vb[i]; cout << endl;

cout << "\tВектор c:"; for(int i=0;i<count;i++) cout<<vc[i]; cout << endl;

}

int \*x1 = new int[count+1]; // Двоичный вектор флагов объединения множеств A и B

int \*x2 = new int[count+1]; // Двоичный вектор флагов пересечения множеств A и B

int \*x3 = new int[count+1]; // Двоичный вектор флагов разности множеств А и B

int \*x4 = new int[count+1]; // Двоичный вектор флагов разности множеств B и A

or(va, vb, x1, count);

and(va, vb, x2, count);

sub(va, vb, x3, count);

sub(vb, va, x4, count);

if(demo!=0){

cout << "\tВектор a|b:"; for(int i=0;i<count;i++) cout<<x1[i]; cout << endl;

cout << "\tВектор a&b:"; for(int i=0;i<count;i++) cout<<x2[i]; cout << endl;

cout << "\tВектор a-b:"; for(int i=0;i<count;i++) cout<<x3[i]; cout << endl;

cout << "\tВектор b-a:"; for(int i=0;i<count;i++) cout<<x4[i]; cout << endl;

}

int \*y = new int[count+1];

xor(c,x1,y, count);

if(sum(y,count)==0) cout << "a|b == c"<< endl;

else cout << "a|b != c"<< endl;

xor(c,x2,y, count);

if(sum(y,count)==0) cout << "a&b == c"<< endl;

else cout << "a&b != c"<< endl;

xor(c,x3,y, count);

if(sum(y,count)==0) cout << "a-b == c"<< endl;

else cout << "a-b != c"<< endl;

xor(c,x4,y, count);

if(sum(y,count)==0) cout << "b-a == c"<< endl;

else cout << "b-a != c"<< endl;

delete y;

delete x1;

delete x2;

delete x3;

delete x4;

delete va;

delete vb;

delete vc;

delete ids;

t = clock()-t;

double seconds = ((double)t)/CLOCKS\_PER\_SEC;

cout << "\tВремя выполнения: " << seconds << "sec" << endl;

delete a;

delete b;

delete c;

getchar();

getchar();

return 0;

}

# Контрольные примеры работы программы

Пример 1.

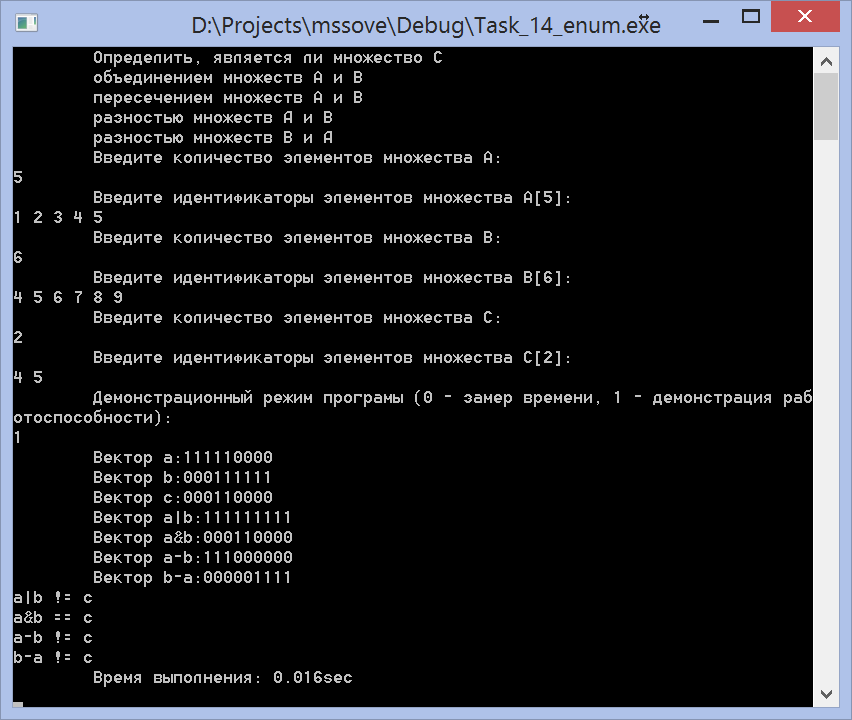


Рис.1..

# Заключение

В ходе данной лабораторной был изучены операции над множествами и булевыми векторами. Была сделана программная реализация данного алгоритма. Для данной программной реализации были проведены ряд тестов, показывающие правильность работы алгоритма.