САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. ПЕТРА ВЕЛИКОГО

Институт прикладной математики и механики

Кафедра «Телематика(при ЦНИИ РТК)»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Генерация Кода Грея и реализация основных операций над множествами.

2 курс, группа 23607/2

Принял:  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Востров А.В.

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016.

Выполнил

Максатбек Б.

Введение

Задание 3

1.Генерация кода Грея 4

2.Выбор массивов двумя способами: Перечисление с клавиатуры и 4 случайным образом

3.Основные операции над множествами 5

4.Протокол тестирования 9

5. Заключение 13

1.Задание

Для введенного с клавиатуры n реализовать генерацию кода Грея(сформировать универсум).

Реализовать выбор массивов двумя способами: перечислением с клавиатуры и случайным образом(мощность задаёт пользователь).

Выполнить операции над множествами – объединение, пересечение, разность, симметрическая разность и дополнение.

2.Генерация кода Грея для заданного n

Определение: **Код Грея** — система счисления, в которой два соседних значения различаются только в одном разряде.

Коды Грея легко получаются из двоичных чисел путём побитовой операции «[Исключающее ИЛИ](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D0%98%D0%9B%D0%98)» с тем же числом, сдвинутым вправо на один бит и в котором старший разряд заполняется нулём. Следовательно, *i*-й бит кода Грея *Gi* выражается через биты двоичного кода *Bi* следующим образом:



Алгоритм на С:

unsigned int grayencode(unsigned int g)

{

**return** g ^ (g >> 1);

}

Реализация:

Генерируем с 0 до максимального возможного(maxSize).

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

int Grey = (i >> 1) ^ i;

U.push\_back(Grey);

for (int j = 0; j < sizeU; j++)

cout << ((Grey & (1 << (sizeU - 1 - j))) > 0);

cout << " " << i << endl;

}

3.Реализация выбора двумя способами

Описание функции:

Функция принимает 5 аргументов: ссылку на вектор, который будет заполняться случайно или в ручную, размер вектора, ссылку на вектор, содержащий универсум, размер универсума и указатель на строку.

Если универсум не равен нулю, то выводим в консоль «Нажмите 'm’ для ручного заполнения и ‘a’ для случайного».  
Пока пользователь не введёт ‘m’ или ‘a’, программа будет просить ввести их.  
Если введено ‘a’, то в вектор случайно добавляются элемент из универсума, пока он не заполнится.

Если введён ‘m’, то вектор будет заполнятся вручную пользователем, пока не будет достигнут максимальный размер вектора.

Если универсум равен нулю, то множества заполняться не будут. В консоли отобразиться «Пустое множество не заполняется».

void Addition(vector<int> &X,int sizeX, vector<int>& U, int maxSize, char \*str)

{

if(sizeX != 0)

{

char c = 0;

cout <<"\nМножество "<< str<<":\n";

cout <<"Нажмите 'm' для ручного заполнения и 'a' для случайного\n";

while (c != 'm' && c != 'a')

{

c = getch();

}

if(c == 'a'){

cout << "Случайное заполнение\n";

for(size\_t i = 0; i < sizeX; i++)

{

int random = rand() % maxSize;

if(incl(X, U[random]) == true)

{

i--;

}

else{

X.push\_back(U[random]);

}

}

}

else {

cout << "Ручное заполнение\n";

for(size\_t i = 0; i < sizeX; i++){

int num;

Input(num, 0,maxSize-1);

cout << endl;

if(incl(X,num) == true){

i--;

cout <<"Такой элемент уже есть\n";

}

else {

X.push\_back(num);

}

}

}

}

else cout << "Пустой массив "<< str<< " не заполняется"<< endl;

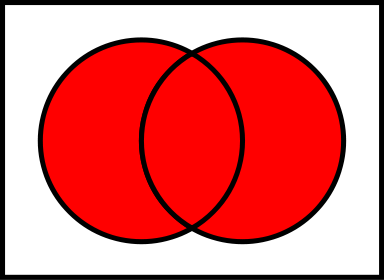
}

4.Основные операции над множествами

1.Объединение

Определение. Объединение - множество, содержащее в себе все элементы исходных множеств.

Для двух множеств

Реализация:  
void Unity(vector<int> &A, vector<int> &B, vector<int> &R){

R.clear();

for(int i = 0; i < A.size();i++)

R.push\_back(A[i]);

for(int i = 0; i < B.size(); i++)

if(!incl(A,B[i]))

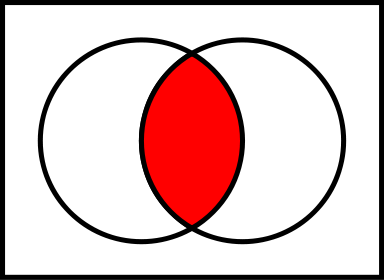
R.push\_back(B[i]);

}

2.Пересечение:

Определение: Пересечение множеств - это [множество](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE), которому принадлежат те и только те [элементы](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE#.D0.AD.D0.BB.D0.B5.D0.BC.D0.B5.D0.BD.D1.82_.D0.BC.D0.BD.D0.BE.D0.B6.D0.B5.D1.81.D1.82.D0.B2.D0.B0), которые одновременно принадлежат всем данным множествам.





Реализация:  
void Cross(vector<int> &B,vector<int> &A, vector<int> &R)

{

R.clear();

for(int i = 0; i < B.size(); i++)

if(incl(A,B[i]))

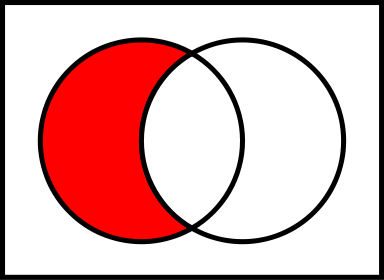
R.push\_back(B[i]);

}

3. Разность:

Разность – множество, в которое входят все элементы первого множества, не входящие во второе множество.





Реализация:

void Diff(vector<int> &X,vector<int> &K, vector<int> &R){

R.clear();

bool t = true;

for(size\_t i = 0; i < X.size(); i++, t = true){

for(size\_t j = 0; j < K.size(); j++){

if(X[i] == K[j])

t = false;

}

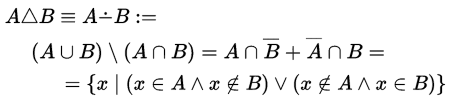
if(t)

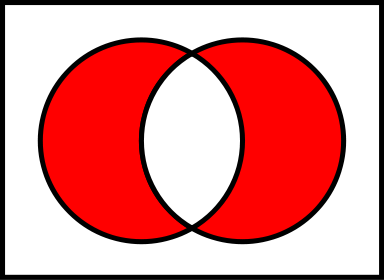
R.push\_back(X[i]);

}

}

4.Симметрическая разность:  
Определение. Симметрическая разность двух [множеств](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE) — теоретико-множественная операция, результатом которой является новое множество, включающее все элементы исходных множеств, не принадлежащие одновременно обоим исходным множествам.





Реализация:

Реализуется через разность.

void sim\_Diff(vector<int> &A,vector<int> &B, vector<int> &R){

R.clear();

bool t = true;

for(size\_t i = 0; i < A.size(); i++, t = true){

for(size\_t j = 0; j < B.size(); j++){

if(A[i] == B[j])

t = false;

}

if(t)

R.push\_back(A[i]);

}

t = true;

for(size\_t i = 0; i < B.size(); i++, t = true){

for(size\_t j = 0; j < A.size(); j++){

if(B[i] == A[j])

t = false;

}

if(t)

R.push\_back(B[i]);

}

}

5.Дополнение  
Операция дополнения подразумевает некоторый зафиксированный универсум (универсальное множество {\displaystyle U}, которое содержит {\displaystyle A}), и сводится к разности множеств с этим универсумом:  


Реализация:

Реализуется через функцию разности. Только вместо множества универсум.  
Diff(U, A, R);

Протокол тестирования

1.Генерация Кода Грея



Рис. 1

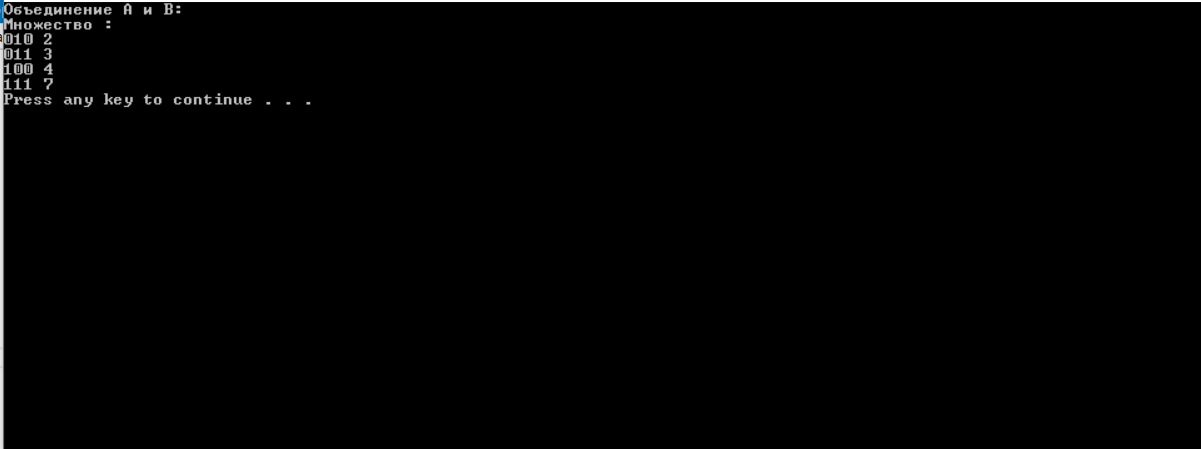
Объединение множеств  


Рис. 2

Пересечение множеств

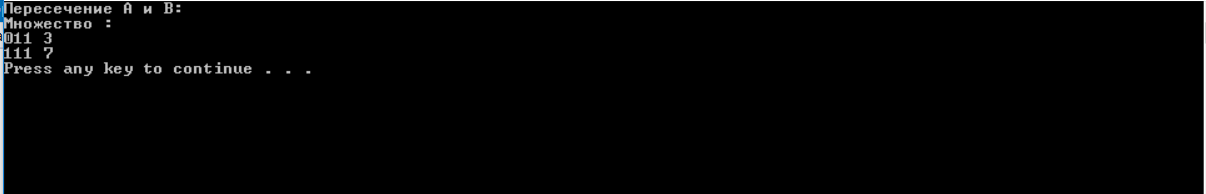


Рис. 3

Разность

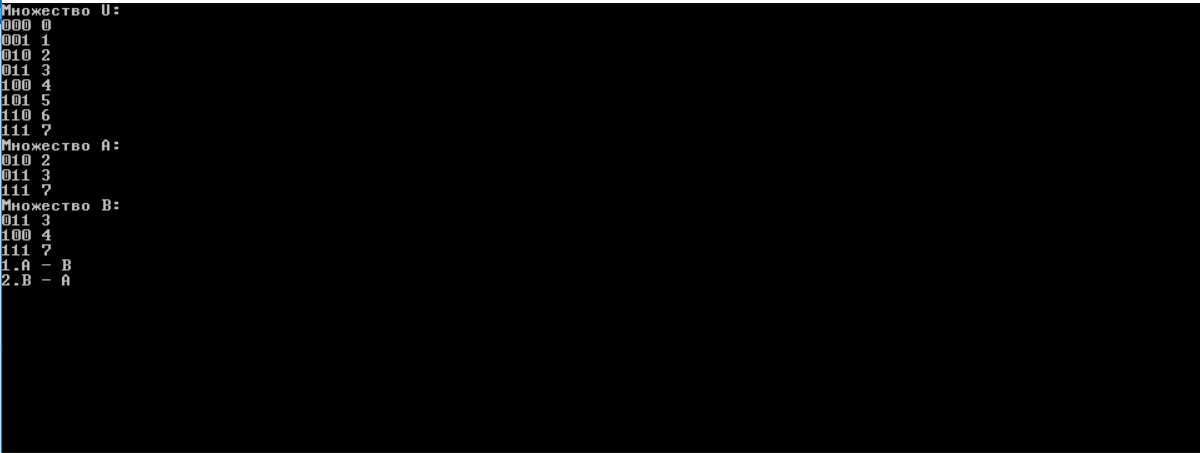


Рис. 4

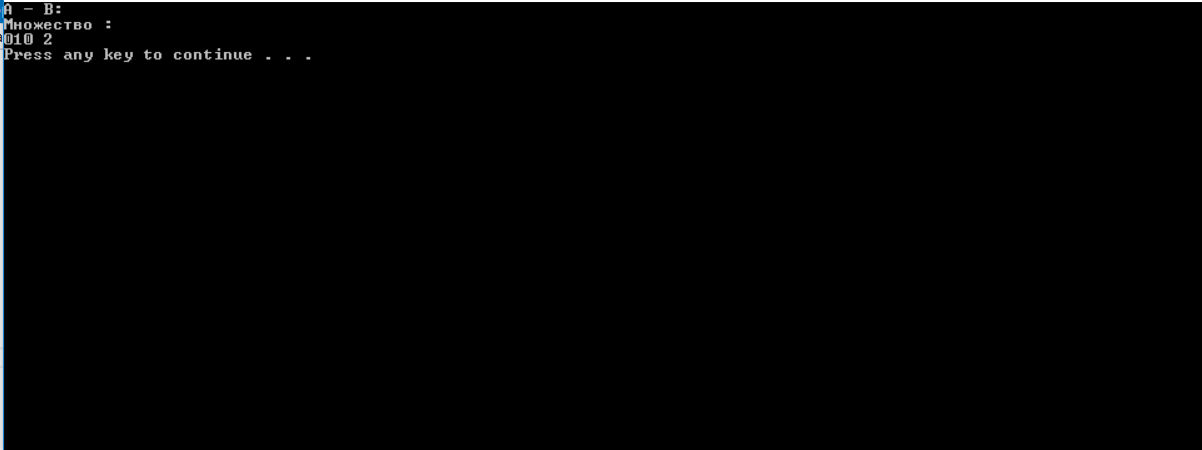


Рис. 5

Симметрическая разность



Рис. 6

Дополнение

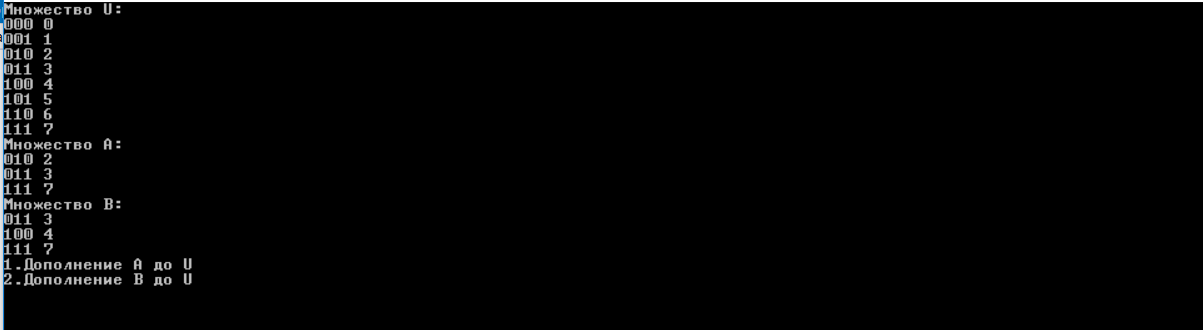


Рис. 7



Рис. 8

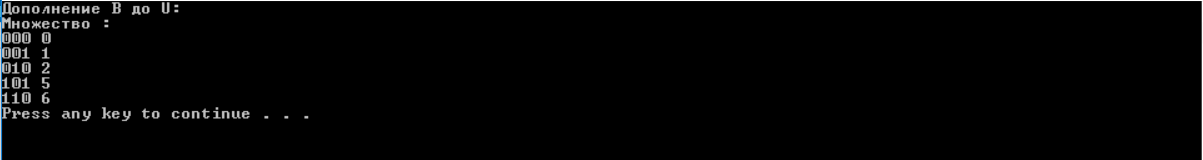


Рис. 9

Переопределение множеств

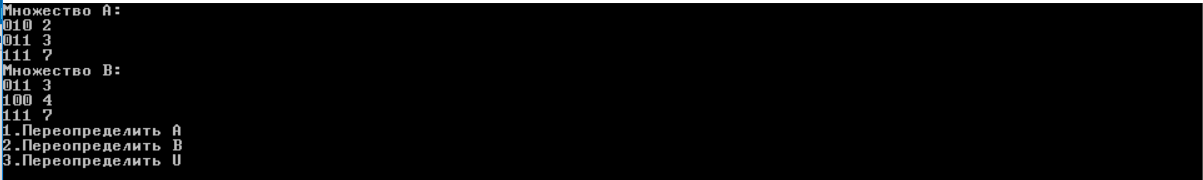


Рис. 10

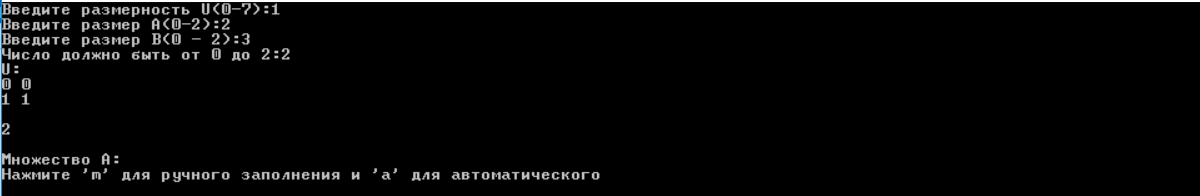


Рис. 11

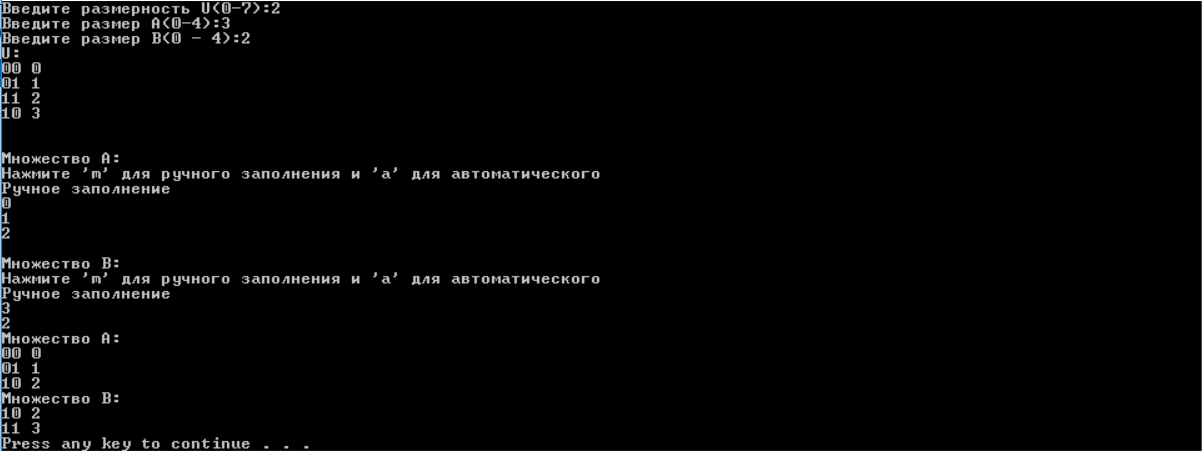


Рис. 12

Некорректный ввод

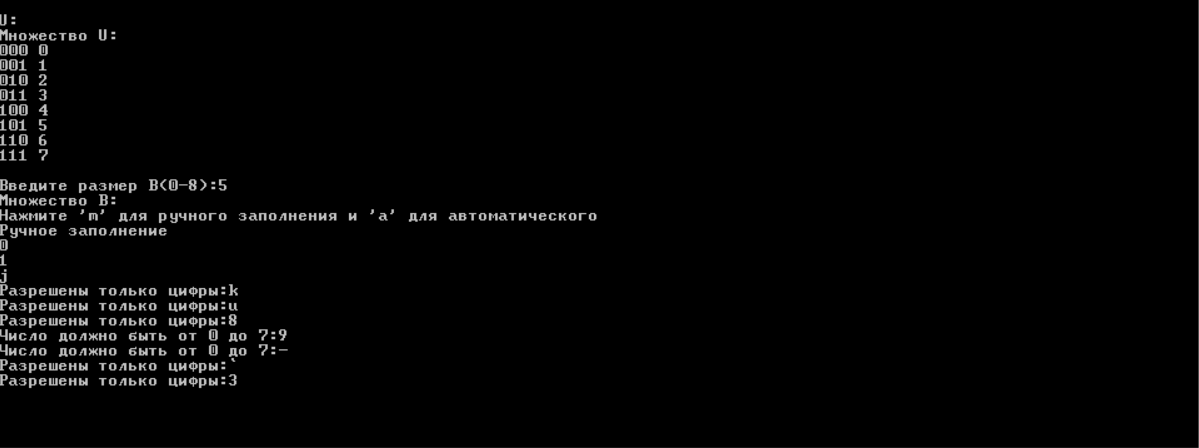


Рис. 13

Универсум 0

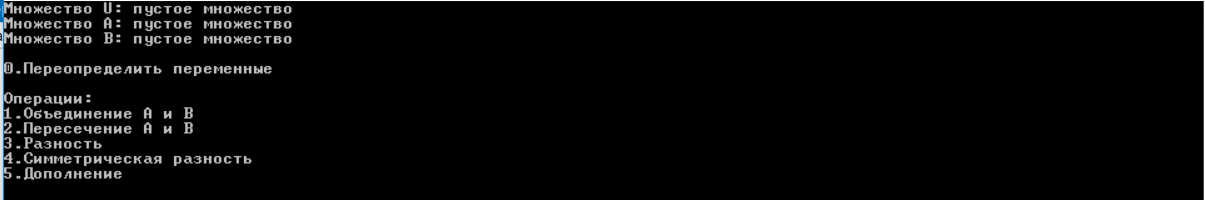


Рис. 14

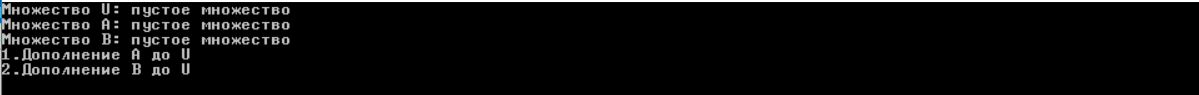


Рис. 15



Рис. 16

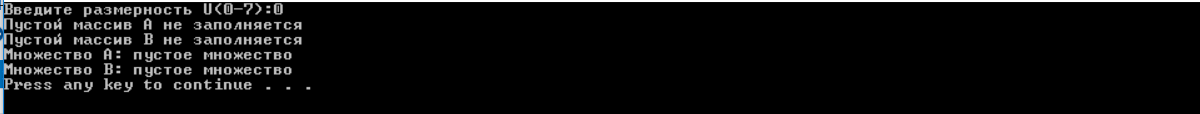


Рис. 17

Заключение

Сгенерирован бинарный код Грея, и основные операции на множествах(объединение, пересечение, разность, симметрическая разность и дополнение) были реализованы.

В ходе выполнения лабораторной работы был получен опыт работы с множествами посредством программирования и применены знания дискретной математике