Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №1

За 3 семестр

По дисциплине «Дискретная математика»

Тема: «Множества»

Выполнил: студент 2 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Глущенко Т.А.

Брест 2020

**Лабораторная работа №1**

**Множества**

**Цель:** изучить базовые понятия теории множеств и операции над ними.

**Вариант 13**

**Задание 1**

На универсуме *U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11}* заданы множества

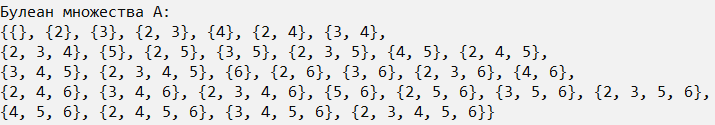
*А = {2, 3, 4, 5, 6}*

*В = {3, 4, 6, 7}*

*С = {3, 5, 8, 9}*

Выражение (АΔВ) ∩С

1. Для заданного множества *A* построить булеан.



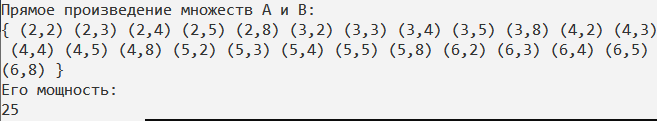
2. На выбор реализовать сортировку *слиянием* либо агоритмом *«слияние»* построить объединение множеств *A* и *B*.



3. Реализовать операцию пересечения множеств, используя маску.

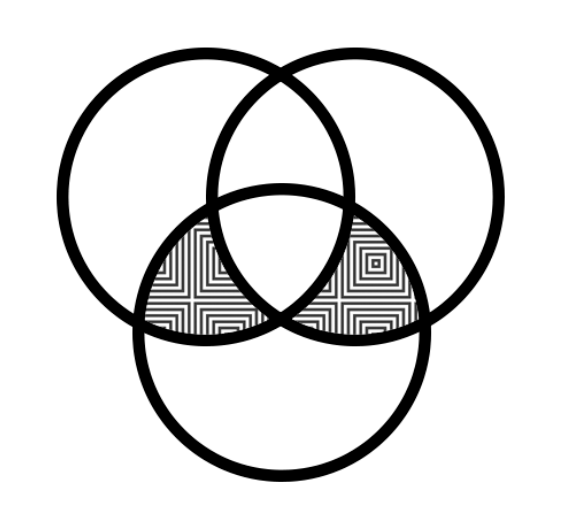


4. Найти прямое произведение  и его мощность.



5. Вычислить программно выражение согласно варианту, проиллюстрировать результат диаграммой Эйлера-Венна.





6. Указать примеры покрытий и разбиения множества *A*.

A = {2, 3, 4, 5, 6}

Разбиения:

E1 = {{2}, {3, 4, 5, 6}}

E2 = {{3}, {2, 4, 5, 6}}

E3 = {{4}, {2, 3, 5, 6}}

E4 = {{5}, {2, 3, 4, 6}}

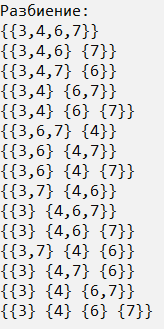
E5 = {{6}, {2, 3, 4, 5}}

Покрытия:

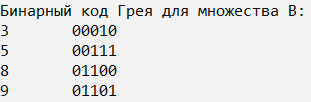
E6 = {{2, 3}, {3, 4, 5, 6}}

E7 = {{3, 4}, {2, 4, 5, 6}}

7. \* Реализовать программно *алгоритм разбиения* множеств.



8. Построить для вашего множества (или множества меньшей мощности) бинарный код *Грея*.



9. Составить таблицу для различных сортировок (быстрой, слиянием, выбором, вставки, пятая на выбор) с указанием *«О большого»*, достоинств и недостатков данного алгоритма сортировки.

|  |  |  |  |
| --- | --- | --- | --- |
| Сортировка | Преимущества | Недостатки | О |
| Быстрая | - при частичной упорядоченности повышаются шансы разделения массива на более равные части  - одна из самых быстрых внутренних сортировок | - использует доп. память  - неустойчивая | Л: O(nlogn)  С: O(nlogn)  Х: O(n2) |
| Слиянием | - устойчивая  - быстрая | - использует доп. память  - если входная последовательность почти отсортирована время работы то же | Л: O(nlogn)  С: O(nlogn)  Х: O(nlogn) |
| Выбором | - устойчивая  - не использует доп. память | - если входная последовательность почти отсортирована, количество сравнений то же | Л: O(n2)  С: O(n2)  Х: O(n2) |
| Вставками | - устойчивая  - эффективна на небольших наборах данных  - эффективна на частично-от-сортированных наборах данных  - не использует доп. память | - высокая вычислительная сложность | Л: O(n)  С: O(n2)  Х: O(n2) |
| Пузырьковая | - простая реализация | - эффективен лишь для небольших массивов  - высокая вычислительная сложность | Л: O(n2)  С: O(n2)  Х: O(n2) |

**Исходный текст программы:**

#include <iostream>

#include <Windows.h>

using namespace std;

void direct\_mul(const int\* A, const int\* B,const int size\_a,const int size\_b) //прямое произведение А и В

{

cout << "Прямое произведение множеств А и В:\n{";

for (int i = 0; i < size\_a; i++) {

for (int j = 0; j < size\_b; j++) {

cout << '(' << A[i] << ',' << B[j] << "), ";

}

}

cout << "\b\b}\nЕго мощность:";

cout << endl << size\_a \* size\_b << endl << endl;

}

void output(int\* U, int size\_u, int x) //пересечение множеств

{

int z = 1 << (size\_u - 1);

for (int i = 0; i < size\_u; i++) {

if ((x & z) == z) {

cout << U[i] << ", ";

}

z = z >> 1;

}

cout << "\b\b}\n";

}

int maska(int\* A, int\* U, int size\_a, int size\_u) //битовая маска

{

int y = 1, x = 0;

for (int i = 0, j = 0; i < size\_a; i++, j++) {

y = 1;

if (A[i] == U[j]) {

y = y << (size\_u - j - 1);

x = x ^ y;

}

else {

i--;

}

}

return x;

}

void maska\_dop(int\* A, int\* B, int\* U, int size\_a, int size\_b, int size\_u) //пересечение множеств, используя маску

{

int x = maska(A, U, size\_a, size\_u);

int y = maska(B, U, size\_b, size\_u);

x = x & y;

cout << "\nПересечение множеств А и В, используя маску = {";

output(U, size\_u, x);

cout << endl;

}

void merge(const int\* A, const int\* B, int const size\_a, int const size\_b) //алгоритм "слияние" для А и В

{

int\* cm = new int[size\_a + size\_b];

int i = 0, j = 0, k = 0;

while (i != size\_a && j != size\_b) {

int data;

if (A[i] < B[j]) {

data = A[i];

i++;

}

else if (A[i] > B[j]) {

data = B[j];

j++;

}

else {

data = A[i];

i++;

j++;

}

cm[k] = data;

k++;

}

if (i == size\_a && j == size\_b);

else {

if (i != size\_a) {

while (i != size\_a) {

cm[k] = A[i];

k++;

i++;

}

}

if (j != size\_b) {

while (j != size\_b) {

cm[k] = B[j];

k++;

j++;

}

}

}

cout << "Объединение множеств А и В алгоритмом \"слияние\":\n" << '{';

for (i = 0; i < k; i++) {

cout << cm[i] << ", ";

}

cout << "\b\b}\n";

}

void Expression(int\* A, int\* B, int\* C, int\* U, int size\_a, int size\_b, int size\_c, int size\_u) //вычисление выражения

{

int x = maska(A, U, size\_a, size\_u);

int y = maska(B, U, size\_a, size\_u);

x = x ^ y;

int c = maska(C, U, size\_c, size\_u);

c = c & x;

cout << "\nВыражение (A" << (char)127 << "B)" << (char)94 << "C = {";

output(U, size\_u, c);

}

void grey\_kod(int\* A, int size\_a) //код Грея

{

int kod[5], tmp;

cout << "\nБинарный код Грея для множества В:" << endl;

for (int i = 0; i < size\_a; i++) {

cout << A[i] << '\t';

tmp = A[i] ^ (A[i] >> 1);

for (int j = 4; j >= 0; j--) {

kod[j] = tmp % 2;

tmp /= 2;

}

for (int c = 0; c < 5; c++) {

cout << kod[c];

}

cout << endl;

}

}

void bylean(int\* A, int size\_a) //булеан множества А

{

double size\_byl = pow(2, size\_a);

int count = 0;

cout << "\nБулеан множества A:\n{{{";

for (int i = 0; i < size\_byl; i++)

{

int number = i;

int j = 0;

cout << '{';

while (number != 0)

{

if (number % 2 == 1) {

cout << A[j] << ", ";

}

number /= 2;

j++;

}

count++;

cout << "\b\b}, ";

if (count == 7) {

cout << endl;

count = 0;

}

}

cout << "\b\b}\n\n";

}

void print(int\* bm, int size\_b, int\* p, int size\_p)

{

int i, j, imax = 1;

for (i = 2; i <= size\_p; i++) {

if (p[i] > imax) {

imax = p[i];

}

}

cout << '{';

for (i = 1; i <= imax; i++) {

cout << "{";

for (j = 1; j <= size\_p; j++) {

if (p[j] == i) {

cout << bm[j] << ",";

}

}

cout << "\b} ";

}

cout << "\b}" << endl;

}

void fragmetation(int\* p, int size\_p, int\* bm, int size\_b, int i = 1, int j = 1)

{

int l;

if (i > size\_b) {

print(bm, size\_b, p, size\_p);

}

else {

for (l = 1; l <= j; l++) {

p[i] = l;

if (l == j) {

fragmetation(p, size\_p, bm, size\_b, i + 1, j + 1);

}

else {

fragmetation(p, size\_p, bm, size\_b, i + 1, j);

}

}

}

}

int main()

{

SetConsoleOutputCP(1251);

system("color F0");

int U[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 },

A[] = { 2, 3, 4, 5, 6 },

B[] = { 3, 4, 6, 7 },

C[] = { 3, 5, 8, 9 };

int size\_u = sizeof(U) / sizeof(\*U); //мощность универса

int size\_a = sizeof(A) / sizeof(\*A); //мощность множества А

int size\_b = sizeof(B) / sizeof(\*B); //мощность множества B

int size\_c = sizeof(C) / sizeof(\*C); //мощность множества C

bylean(A, size\_a);

direct\_mul(A, B, size\_a, size\_b);

merge(A, B, size\_a, size\_b);

Expression(A, B, C, U, size\_a, size\_b, size\_c, size\_u);

maska\_dop(A, B, U, size\_a, size\_b, size\_u);

grey\_kod(C, size\_c);

int pm[100];

int\* p = new int[size\_b + 1];

cout << "\n" << "Разбиение: \n";

for (int j = 1; j <= size\_b; j++) {

pm[j] = B[j - 1];

}

fragmetation(p, size\_b + 1, pm, size\_b);

return 0;

}

**Задание 2**

В группе из *100* студентов *35* изучают французский язык, *42* - испанский, *43* - немецкий, *17* изучают французский и испанский, *15* - испанский и немецкий, *13* - французский и немецкий, и *20* студентов не изучают ни один из трех языков.

а) Сколько студентов изучают французский или немецкий язык, но не изучают испанский?

б) Сколько студентов изучают только один из трех языков?

в) Сколько студентов изучают два из трех языков?

г) Сколько студентов не изучают ни испанский язык, ни французский?

д) Сколько студентов изучают только испанский язык?

А – 38 Б – 45 В – 30 Г – 30 Д – 15

