**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение высшего образования

**«Национальный исследовательский ядерный университет МИФИ»**

Обнинский институт атомной энергетики – филиал НИЯУ МИФИ

**Отделение интеллектуальных кибернетических систем**

Объектно-ориентированное программирование

ЛАБОРАТОРНАЯ РАБОТА №1

«Мультимножество»

Выполнил: студент

 группы М-М20

Богомягкова А.С.

Руководитель:

Тельнов В.П.

Обнинск 2021г.

**1.Постановка задачи**

Разработать в MS Visual Studio программное решение на языке Си, которое реализует динамическую структуру данных (контейнер) типа «Мультимножество». Здесь под «мультимножеством» понимается упорядоченное множество элементов, которые не обязаны быть уникальными (т.е. могут повторяться). Каждый элемент контейнера содержит строки символов произвольной длины.

В программном решении следует реализовать следующие операции над контейнером:

* создание и уничтожение контейнера;
* добавление, извлечение и поиск элементов контейнера;
* обход всех элементов контейнера (итератор);
* вычисление количества элементов в контейнере;
* объединение, пересечение и вычитание контейнеров;
* сохранение контейнера в дисковом файле и восстановление контейнера из файла.

Письменный отчет по работе должен содержать следующие разделы:

1. Постановку задачи.
2. Описание контейнера как динамической структуры данных, в том числе:
* рисунки, на которых изображена структура данных и поясняются основные операции над ней;
* описание алгоритмов, которые используются при работе с контейнером;
* область применения данной структуры данных, её преимущества и недостатки в сравнении с другими известными структурами.
1. Структурную схему программы, на которой показаны все реализованные функции.
2. Руководство пользователя (как пользоваться программой, возможные ошибки).
3. Структуру папок разработанного проектного решения.
4. Листинг разработанного авторского кода на языке Cи. Код должен быть надлежащим образом структурирован и снабжен комментариями.

**2. Описание контейнера как динамической структуры данных**

Мультимножество — модификация понятия множества, допускающая включение одного и того же элемента в совокупность по нескольку раз. Число элементов в мультимножестве, с учётом повторяющихся элементов, называется его размером или мощностью.

Операции, связанные с этим типом данных, позволяют выполнять:

* добавление элементов в коллекцию
* удаление элементов из коллекции
* проверить наличие элемента в мультимножестве
* получить размер мультимножества

**2.1 Рисунки, на которых изображена структура данных и поясняются основные операции над ней**

* Структура множества



* Добавление элементов в коллекцию



* Удаление элементов из коллекции



**2.2 Описание алгоритмов, которые используются при работе с контейнером**

* 1. Создание контейнера:

Выделяем память в контейнер с помощью оператора new, задаем значение переменной длинны значение равное 0 и устанавливаем нулевой указатель на массив строк.

* 1. Уничтожение контейнера:

С помощью оператора delete которая возвращает память в кучу. Устанавливаем нулевой указатель на массив строк.

* 1. Добавление элемента:

Вводим строку символов элемента контейнера. Каждый введенный символ записывается в массив типа char. Ввод символов происходит до тех пор, пока не достигли конца строки, то есть не нажмем «enter». После нажатия «enter» выделяем память для нового элемента и добавляем его в массив строк и увеличиваем значение переменной длинны массива на 1.

* 1. Удаление элемента

Вводим строку. Проходим по всем элементам контейнера и сравниваем значения введенной строки и строк в контейнере. Если такая строка найдена, то сдвигаем влево каждый элемент, находящийся справа от удаляемого элемента. Сокращаем память на единицу и уменьшаем значение переменной длинны массива на 1.

* 1. Вывод элементов контейнера

Если контейнер не пуст проходим по всем элементам контейнера и последовательно выводим с помощью функции printf() значения строк.

* 1. Пересечение контейнеров

Необходимо создать два контейнера. Проходим по значениям первого и второго контейнеров. Если находим одинаковые значения, то выделяем (для первого элемента контейнера), а затем расширяем динамическую память для нового контейнера и записываем в новый контейнер этот элемент.

7. Разность контейнеров

Необходимо создать два контейнера. Проходим по значениям первого и второго контейнеров. Если находим уникальные элементы в первом множестве то выделяем (для первого элемента контейнера), а затем расширяем динамическую память для нового контейнера и записываем в новый контейнер этот элемент.

8. Объединение контейнеров

Необходимо создать два контейнера. Выделяем (для первого элемента контейнера), а затем расширяем динамическую память для нового контейнера и записываем в новый контейнер значения строк первого контейнера. Далее после проверки на уникальность строк расширяем динамическую память нового контейнера и из второго контейнера записываем в новый контейнер значения строк из второго контейнера.

9. Сохранения контейнера в дисковом файле

Если контейнер не пустой открываем файл для записи данных (Путь и имя файла необходимо ввести). Делаем проверку файла на открытие. Проходим по всем элементам контейнера и последовательно, с помощью функции fprintf(), записываем значение в файл. Закрываем файл.

1. Восстановление контейнера из файла

Открываем файл по пути вписанном пользователем. Делаем проверку файла на открытие. Пока не достигнут конец файла, выделяем (для первого контейнера), а затем увеличивается на ячейку (для контейнера) динамическую память для нового контейнера. Перед записью в контейнер необходимо его очистить. Считываем из файла строку во временную переменную. Для элемента контейнера «строка» выделяем динамически память в соответствии с размером строки в файле. Присваиваем элементу контейнера «строка» строку из файла.

1. Проверка на уникальность

Передаем функции значение строки. Если значение строки не равно ни одному значению строки из контейнера, то строка уникальна.

**2.3 Область применения данной структуры данных**

Тип данных мультимножество применяют для удобства работы с математическими мультимножествами, так как программа может реализовать все возможные операции над мультимножествами гораздо быстрее, чем это сделает человек.

**3. Структурная схема программы, на которой показаны все реализованные функции**



**4. Руководство пользователя (как пользоваться программой, возможные ошибки).**

Для начала работы с программой необходимо запустить файл. Появится окно консольного приложения со списком возможных операций на русском языке. Слева от операции указывается соответствующая клавиша («1» - «17», «0»).

При запуске открывается окно с четырьмя пунктами:



 «1», «2» - создание первого и второго контейнера соответственно. На экран выводится «Мультимножество создано».

«0» - выход.

После выполнения пункта 1 появляются дополнительные возможности:



Где:

«3» - восстановление мультимножества из файла. Необходимо ввести путь до файла. Если файл найден, на экран будет выведено что файл открыт. При возникновении ошибки на экран будет выведена ошибка.

«4» - добавление элемента в первый контейнер. Выводится сообщение «Введите элемент».

«5» - удаление элементов из первого контейнера. На экран выводится «Введите элемент». В случае если элемент не найден выводится «Данный элемент не найден».

«6» - вывод первого контейнера. Если количество элементов контейнера равно нулю, то выводится «Мультимножество пустое».

«7» - подсчет элементов в первом контейнере. На экран выводится «Количество элементов в первом/втором мультимножестве: (число элементов)».

«8» - очистка памяти, выделенная под первый контейнер.

«9» - сохранение первого мультимножества в файл. Путь до файла необходимо ввести. При возникновении ошибки, на экран будет выведена ошибка.

После выполнения 2 действия:



Где:

Пункты «10» - «16» - аналогичны пунктам «3» - «9» только действия производятся со вторым мультимножеством.

После заполнения обоих контейнеров хотя бы одним элементом будут доступны следующие действия:



«17» - пересечение двух контейнеров. На экран выводится три мультимножества. Первая строка: первое мультимножество; вторая строка: второе мультимножество; третья строка: мультимножество, полученное при пересечении.

«18», «19» - разность мультимножеств первого со вторым и второго с первым соответственно. Для вычисления разности необходимо создать два контейнера. На экран выводится три мультимножества. Первая строка: первое мультимножество; вторая строка: второе мультимножество; третья строка: мультимножество, полученное при разности.

«20» - объединение двух контейнеров. Перед объединением необходимо создать два контейнера. На экран выводится три мультимножества. Первая строка: первое мультимножество; вторая строка: второе мультимножество; третья строка: мультимножество, полученное при пересечении.

Так как действия «17» - «20»- производят запись в третий контейнер. После выполнения одного из действий будут доступны новые действия:



Где:

«21» - вывод всех элементов третьего мультимножества.

«22» - вывод количества элементов третьего мультимножества.

«23» - очистка третьего контейнера.

«24» -сохранение третьего контейнера в файл.

**5. Структура папок разработанного проектного решения.**



**6. Листинг разработанного авторского кода на языке Си**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

using namespace std;

typedef struct MyStruct

{

 int count;

 char\*\* mass;

} SET;

int Quantity(char\*\* mass, int count, char\* el)

{

 int iter = 0;

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

 {

 if (strcmp(mass[i], el) == 0)

 {

 iter++;

 }

 }

 return iter;

}

SET\* Create(SET\* mn)

{

 mn = new SET;

 mn->mass = NULL;

 mn->count = 0;

 return mn;

}

SET\* Clearing(SET\* mn)

{

 if (mn->mass != NULL)

 {

 delete[] mn->mass;

 }

 delete mn;

 mn = NULL;

 return mn;

}

int Unique(char\*\* mass, int count, char\* el)

{

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

 {

 if (strcmp(mass[i], el) == 0)

 {

 return i;

 }

 }

 return -1;

}

SET\* ADD(SET\* mn)

{

 char\* el = new char;

 printf("Введите элемент\n");

 scanf("%s", el);

 if (mn->mass == NULL)

 {

 char\*\* newmass = new char\*[1];

 newmass[0] = el;

 mn->mass = newmass;

 mn->count += 1;

 }

 else

 {

 char\*\* newmass = new char\*[mn->count + 1];

 for (int i = 0; i < mn->count; i++)

 {

 newmass[i] = mn->mass[i];

 }

 newmass[mn->count] = el;

 delete[] mn->mass;

 mn->mass = newmass;

 mn->count += 1;

 }

 return mn;

}

SET\* Extraction(SET\* mn)

{

 char\* el = new char;

 printf("Введите элемент\n");

 scanf("%s", el);

 int iter = Unique(mn->mass, mn->count, el);

 if (iter != -1)

 {

 if (mn->count != 1)

 {

 char\*\* newmass = new char\*[mn->count - 1];

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

 {

 newmass[i] = mn->mass[i];

 }

 for (int j = iter + 1; j < mn->count; j++)

 {

 newmass[j - 1] = mn->mass[j];

 }

 delete[] mn->mass;

 mn->mass = newmass;

 mn->count -= 1;

 }

 else

 {

 delete[] mn->mass;

 mn->count = 0;

 mn->mass = NULL;

 }

 }

 else

 {

 printf("Данного элемента в мультимножестве не обнаружено\n");

 }

 return mn;

}

void Out(SET\* mn)

{

 int c = mn->count;

 if (c != 0)

 {

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

 {

 printf(" %s", mn->mass[i]);

 }

 }

 else

 {

 printf("Пустое множество");

 }

}

SET\* Intersections(SET\* mn1, SET\* mn2, SET\* mn3)

{

 int c = 0;

 char\*\* newmass3 = new char\*[mn3->count];

 int fl = 0;

 for (int i = 0; i < mn1->count; i++)

 {

 int mn1quant = Quantity(mn1->mass, mn1->count, mn1->mass[i]);

 int mn2quant = Quantity(mn2->mass, mn2->count, mn1->mass[i]);

 if (mn2quant != 0 && Unique(newmass3, c, mn1->mass[i]) == -1)

 {

 int ss;

 if (mn1quant < mn2quant)

 {

 ss = mn1quant;

 }

 else

 {

 ss = mn2quant;

 }

 char\*\* newmass2 = new char\*[c + ss];

 for (int k = 0; k < c; k++)

 {

 newmass2[k] = newmass3[k];

 }

 for (int k = c; k < c + ss; k++)

 {

 newmass2[k] = mn1->mass[i];

 }

 delete[] newmass3;

 newmass3 = newmass2;

 c += ss;

 }

 }

 for (int i = 0; i < mn2->count; i++)

 {

 int mn1quant = Quantity(mn1->mass, mn1->count, mn2->mass[i]);

 int mn2quant = Quantity(mn2->mass, mn2->count, mn2->mass[i]);

 if (mn1quant != 0 && Unique(newmass3, c, mn2->mass[i]) == -1)

 {

 int ss;

 if (mn1quant < mn2quant)

 {

 ss = mn1quant;

 }

 else

 {

 ss = mn2quant;

 }

 char\*\* newmass2 = new char\*[c + ss];

 for (int k = 0; k < c; k++)

 {

 newmass2[k] = newmass3[k];

 }

 for (int k = c; k < c + ss; k++)

 {

 newmass2[k] = mn2->mass[i];

 }

 delete[] newmass3;

 newmass3 = newmass2;

 c += ss;

 }

 }

 char\*\* newmass1 = new char\*[mn3->count + c];

 for (int i = 0; i < mn3->count; i++)

 {

 newmass1[i] = mn3->mass[i];

 }

 for (int i = mn3->count; i < mn3->count + c; i++)

 {

 newmass1[i] = newmass3[i - mn3->count];

 }

 c += mn3->count;

 delete[] mn3->mass;

 mn3->count = c;

 mn3->mass = newmass1;

 return mn3;

}

SET\* Difference(SET\* mn1, SET\* mn2, SET\* mn3)

{

 int c = 0;

 char\*\* newmass3 = new char\*[mn3->count];

 int iter;

 char\* el;

 int mn1iter = 0;

 int mn2iter = 0;

 int quan = 0;

 for (int i = 0; i < mn1->count; i++)

 {

 iter = 0;//зануляем

 mn1iter = Quantity(mn1->mass, mn1->count, mn1->mass[i]);

 mn2iter = Quantity(mn2->mass, mn2->count, mn1->mass[i]);

 quan = abs(mn1iter - mn2iter);

 if (mn1iter > mn2iter && Unique(newmass3, c, mn1->mass[i]) == -1)

 {

 char\*\* newmass2 = new char\*[c + quan];

 for (int k = 0; k < c; k++)

 {

 newmass2[k] = newmass3[k];

 }

 for (int j = c; j < c + quan; j++)

 {

 newmass2[j] = mn1->mass[i];;

 }

 delete[] newmass3;

 newmass3 = newmass2;

 c = c + quan;

 }

 }

 char\*\* newmass1 = new char\*[mn3->count + c];

 for (int i = 0; i < mn3->count; i++)

 {

 newmass1[i] = mn3->mass[i];

 }

 for (int i = mn3->count; i < mn3->count + c; i++)

 {

 newmass1[i] = newmass3[i - mn3->count];

 }

 c += mn3->count;

 delete[] mn3->mass;

 mn3->count = c;

 mn3->mass = newmass1;

 return mn3;

}

SET\* Union(SET\* mn1, SET\* mn2, SET\* mn3)

{

 mn3 = Difference(mn1, mn2, mn3);

 mn3 = Intersections(mn1, mn2, mn3);

 mn3 = Difference(mn2, mn1, mn3);

 return mn3;

}

int SaveInFile(SET\* mn)

{

 if (mn != NULL)

 {

 if (mn->mass != NULL)

 {

 char\* name = new char;

 printf("Введите имя файла для сохраниения\n");

 printf("Пример: D:\\sets.txt\n");

 scanf("%s", name);

 FILE\* file;

 file = fopen(name, "w");

 if (file != NULL)

 {

 printf("Файл открыт\n");

 for (int i = 0; i < mn->count; i++)

 {

 fprintf(file, " %s", mn->mass[i]);

 }

 fclose(file);

 return 1;

 }

 else

 {

 printf("Файл ненайден\n");

 return 0;

 }

 }

 else

 {

 printf("Мультимножество пустое\n");

 return 0;

 }

 }

 else

 {

 printf("Контейнер мультимножества не создан\n");

 }

 return 0;

}

SET\* ReadFromFile(SET\* mn)

{

 char\* name = new char;

 int c = 0;

 char\*\* newmass1 = new char\*[c];

 printf("Введите имя файла для сохраниения\n");

 printf("Пример: D:\\sets.txt\n");

 scanf("%s", name);

 FILE\* file;

 file = fopen(name, "r");

 if (file != NULL)

 {

 printf("Файл открыт\n");

 while (!feof(file))

 {

 char\*\* newmass2 = new char\*[c + 1];

 char\* el = new char;

 for (int k = 0; k < c; k++)

 {

 newmass2[k] = newmass1[k];

 }

 fscanf(file, " %s", el);

 newmass2[c] = el;

 delete[] newmass1;

 newmass1 = newmass2;

 c++;

 }

 delete[] mn->mass;

 mn->count = c;

 mn->mass = newmass1;

 printf("Мультимножество восстановленно\n");

 return mn;

 }

 else

 {

 printf("Файл ненайден\n");

 return mn;

 }

}

void main()

{

 setlocale(LC\_ALL, "Russian");

 int q = 1;

 int c;

 int in;

 SET\* mn1 = NULL;

 SET\* mn2 = NULL;

 SET\* mn3 = NULL;

 while (q)

 {

 if (mn1 == NULL)

 printf("1)Создание первого мультимножества\n");

 if (mn2 == NULL)

 printf("2)Создание второго мультимножества\n");

 if (mn1 != NULL)

 {

 printf("3)Восставновление мультимножества из файла в первый контейнер\n");

 printf("4)Добавление элемента в первое мультимножество\n");

 printf("5)Удаление элемента из первого мультимножества\n");

 printf("6)Обход всех элементов первого мультимножества\n");

 printf("7)Количество элементов первого мультимножества\n");

 printf("8)Очистка первого контейнера\n");

 printf("9)Сохранение первого мультимножества в файл\n");

 }

 if (mn2 != NULL)

 {

 printf("10)Восставновление мультимножества из файла во второй контейнер\n");

 printf("11)Добавление элемента во второе мультимножество\n");

 printf("12)Удаление элемента из второго мультимножества\n");

 printf("13)Обход всех элементов второго мультимножества\n");

 printf("14)Количестов элементов второго мультимножества\n");

 printf("15)Очистка второго контейнера\n");

 printf("16)Сохранение второго мультимножества в файл\n");

 }

 if (mn1 != NULL && mn2 != NULL)

 if (mn1->count != 0 && mn2->count != 0)

 {

 printf("17)Пересечение первого и второго мультимножества\n");

 printf("18)Разность первого и второго мультимножества\n");

 printf("19)Разность второго и первого мультимножества\n");

 printf("20)Объединение первого и второго мультимножества\n");

 }

 if (mn3 != NULL)

 if (mn3->count != 0)

 {

 printf("21)Обход всех элементов третьего мультимножества\n");

 printf("22)Количестов элементов третьего мультимножества\n");

 printf("23)Очистка третьего контейнера\n");

 printf("24)Сохранение третьего мультимножества в файл\n");

 }

 printf("0)Закончить работу\n");

 printf("Введите цифру действия \n");

 scanf("%d", &c);

 switch (c)

 {

 case 1:

 if (mn1 == NULL)

 {

 mn1 = Create(mn1);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 2:

 if (mn2 == NULL)

 {

 mn2 = Create(mn2);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 3:

 if (mn1 != NULL)

 {

 mn1 = ReadFromFile(mn1);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 4:

 if (mn1 != NULL)

 {

 mn1 = ADD(mn1);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 5:

 if (mn1 != NULL)

 {

 mn1 = Extraction(mn1);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 6:

 if (mn1 != NULL)

 {

 printf("Ваше мультимножество: ");

 Out(mn1);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 7:

 if (mn1 != NULL)

 {

 printf("Колличество элементов в контейнере: %d\n", mn1->count);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 8:

 if (mn1 != NULL)

 {

 mn1 = Clearing(mn1);

 if (mn1 == NULL)

 {

 printf("Память очищена\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 9:

 if (mn1 != NULL)

 {

 in = SaveInFile(mn1);

 if (in == 1)

 {

 printf("Мультимножество сохранено\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 10:

 if (mn2 != NULL)

 {

 mn2 = ReadFromFile(mn2);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 11:

 if (mn2 != NULL)

 {

 mn2 = ADD(mn2);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 12:

 if (mn2 != NULL)

 {

 mn2 = Extraction(mn2);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 13:

 if (mn2 != NULL)

 {

 printf("Ваше множество\n");

 Out(mn2);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 14:

 if (mn2 != NULL)

 {

 printf("Колличество элементов в контейнере: %d\n", mn2->count);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 15:

 if (mn2 != NULL)

 {

 mn2 = Clearing(mn2);

 if (mn2 == NULL)

 {

 printf("Память очищена\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 16:

 if (mn2 != NULL)

 {

 in = SaveInFile(mn2);

 if (in == 1)

 {

 printf("Мультимножество сохранено\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 17:

 if (mn1 != NULL && mn2 != NULL)

 {

 if (mn1->count != 0 && mn2->count != 0)

 {

 if (mn3 != NULL)

 {

 mn3 = Clearing(mn3);

 }

 mn3 = Create(mn3);

 mn3 = Intersections(mn1, mn2, mn3);

 printf("Первое мультимножество:");

 Out(mn1);

 printf("\n");

 printf("Второе мультимножество:");

 Out(mn2);

 printf("\n");

 printf("Пересечение: ");

 Out(mn3);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 18:

 if (mn1 != NULL && mn2 != NULL)

 {

 if (mn1->count != 0 && mn2->count != 0)

 {

 if (mn3 != NULL)

 {

 mn3 = Clearing(mn3);

 }

 mn3 = Create(mn3);

 mn3 = Difference(mn1, mn2, mn3);

 printf("Первое мультимножество:");

 Out(mn1);

 printf("\n");

 printf("Второе мультимножество:");

 Out(mn2);

 printf("\n");

 printf("Разность: ");

 Out(mn3);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 19:

 if (mn1 != NULL && mn2 != NULL)

 {

 if (mn1->count != 0 && mn2->count != 0)

 {

 if (mn3 != NULL)

 {

 mn3 = Clearing(mn3);

 }

 mn3 = Create(mn3);

 mn3 = Difference(mn2, mn1, mn3);

 printf("Первое мультимножество:");

 Out(mn1);

 printf("\n");

 printf("Второе мультимножество:");

 Out(mn2);

 printf("\n");

 printf("Разность: ");

 Out(mn3);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 20:

 if (mn1 != NULL && mn2 != NULL)

 {

 if (mn1->count != 0 && mn2->count != 0)

 {

 if (mn3 != NULL)

 {

 mn3 = Clearing(mn3);

 }

 mn3 = Create(mn3);

 mn3 = Union(mn1, mn2, mn3);

 printf("Первое мультимножество:");

 Out(mn1);

 printf("\n");

 printf("Второе мультимножество:");

 Out(mn2);

 printf("\n");

 printf("Объединение: ");

 Out(mn3);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 21:

 if (mn3 != NULL)

 {

 if (mn3->count != NULL)

 {

 printf("Ваше мультимножество: ");

 Out(mn3);

 printf("\n");

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 22:

 if (mn3 != NULL)

 {

 if (mn3->count != NULL)

 {

 printf("Колличество элементов в контейнере: %d\n", mn3->count);

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 23:

 if (mn3 != NULL)

 {

 if (mn3->count != NULL)

 {

 mn3 = Clearing(mn3);

 if (mn3 == NULL)

 {

 printf("Память очищена\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 24:

 if (mn3 != NULL)

 {

 if (mn3->count != NULL)

 {

 in = SaveInFile(mn3);

 if (in == 1)

 {

 printf("Мультимножество сохранено\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 }

 else

 {

 printf("Действие не доступно\n");

 }

 break;

 case 0:

 q = 0;

 if (mn1 != NULL)

 mn1 = Clearing(mn1);

 if (mn2 != NULL)

 mn2 = Clearing(mn2);

 if (mn3 != NULL)

 mn3 = Clearing(mn3);

 break;

 }

 }

}