#include <iostream>

#include <string>

#include <iomanip>

#include <fstream>

#include <sstream>

using namespace std;

//Cтудент группы

struct Student

{

string name; //ФИО студента

int marks[5];//последние пять оценок за сессию

int studentship;//размер стипендии

Student\* next;

Student\* prev;

};

//Группа потока

struct Group

{

int GroupNumebr; //номер группы

Student\* FirstStudent;//указатель на первого студента группы

Group\* next;

Group\* prev;

};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//////////ФУНКЦИИ////////////////////////////////////////////////

Student\* CreateStudent(int \_tempGroupNumber);//создание студента

Group\* CreateGroup(int \_tempGroupNumber);// создание новой группы

Group\* SearchGroup(Group\* X, int \_tempGroupNumber);//поиск группы

Student\* SearchStudent(Student\* x, string \_tempFIO);//поиск студента

Group\* SortGroups(Group\* Head);// Сортировка списка групп во возрастанию номера группы

void PrintStudent(Student\* x, int groupNumber);//печать данных о студенте

void PrintGroup(Group\* X);//печать данных о группе

void DeleteStudent(Student\* x);//удаление студента из группы

void DeleteGroup(Group\* X);//удаление группы из потока

void DeleteAll(Group\* Head);//удаление всей информации

void HeadOfPrint();//вывод шапки таблицы для вывода данных

// ФУНКЦИИ ГЛАВНОГО МЕНЮ //

void MainMenu(Group\* Head);//вывод на консоль главного меню

//Ввод данных

Group\* AddStudent(Group\* X);//добавление студента в созданную группу

Group\* AddGroup(Group\* X);//добавление новой группы

//Вывод данных

void PrintStudent\_MainMenuFunction(Group\* X); //вывод данных о студенте

void PrintGroup\_MainMenuFunction(Group\* X); //вывод данных о группе

void PrintAll(Group\* X);//вывод данных о потоке

//Редактирование данных о студенте

Group\* RedactInformationOfStudent(Group\* Head);

//Сохранение в файл

void SaveToFile(string filename, Group\* Head);

//Загрузка из файла

Group\* AddinFile(string filename, Group\* Head);

//Удаление данных

void DeleteGroup\_MainMenuFunction(Group\* Head); //удаление информации о группе студентов

void DeleteStudent\_MainMenuFunction(Group\* Head); //удаление информации о студенте группы

// Персональное задание номер 4:

void PersonalTuck(Group\* Head);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*Это базовые функции программы, они используются внутри функций главного меню\*/

//Создание студента

//Создание узла в списке студентов

//Возвращает указатель на новый узел

Student\* CreateStudent(int \_tempGroupNumber)

{

Student\* x = new Student;

string F;

string I;

string O;

string FIO;

do

{

cout << "Введите ФИО студента(не более 50 символов). Имя и отчество вводите полностью. " << endl;;

cout << "Введите Фамилию: "; cin >> F;

cout << "Введите Имя: "; cin >> I;

cout << "Введите Отчество: "; cin >> O;

FIO = F + " " + I + " " + O;

if (FIO.size() > 50)

{

cout << "Количество символов в поле ФИО превышает допустимый лимит в 50 символов. Пожалуйста, введите данные еще раз." << endl;

FIO.clear();

}

} while (FIO.empty());

x->name = FIO;

cout << "Введите последние пять оценок за сессию (без названий предметов)." << endl;

for (int i = 0; i < 5; i++)

{

x->marks[i] = 0;

}

int \_tempMark;

for (int i = 0; i < 5; i++)

{

while (x->marks[i] == 0)

{

cout << "Оценка " << i + 1 << ": ";

cin >> \_tempMark;

if (\_tempMark >= 2 && \_tempMark <= 5)

{

x->marks[i] = \_tempMark;

}

else

{

cout << "Оценка не может быть меньше двух или больше пяти. Введите данные еще раз: ";

}

}

}

cout << "Введите размер стипендии студента: ";

cin >> x->studentship;

x->next = nullptr;

x->prev = nullptr;

return x;

}

//Создание группы

//Создает новый узел списка группы, возвращает указатель на него

Group\* CreateGroup(int \_tempGroupNumber)

{

Group\* X = new Group;

X->GroupNumebr = \_tempGroupNumber;

Student\* temp = new Student;

temp = CreateStudent(\_tempGroupNumber);

X->FirstStudent = temp;

int action;

do

{

cout << "Студент добавлен в группу. Хотите ли вы добавить еще ?" << endl;

cout << "1 - да, любое другое значение - нет. ";

cin >> action;

if (action == 1)

{

system("cls");

temp->next = CreateStudent(\_tempGroupNumber);

temp->next->prev = temp;

temp = temp->next;

}

} while (action == 1);

X->next = nullptr;

X->prev = nullptr;

return X;

}

//Поиск группы

//Возвращает указатель на узел списка, если совпадают номера группы

//Если нет, возвращает NULL

Group\* SearchGroup(Group\* X, int \_tempGroupNumber)

{

while (X != nullptr)

{

if (X->GroupNumebr == \_tempGroupNumber)

{

break;

}

X = X->next;

}

return X;

}

//Поиск студента

//Возвращает указатель на узел списка, если совпадают строки ФИО

//Если нет, возвращает NULL

Student\* SearchStudent(Student\* x, string \_tempFIO)

{

while (x != nullptr)

{

if (x->name == \_tempFIO)

{

break;

}

x = x->next;

}

return x;

}

//Сортировка списка групп по возрастанию номеров группы

//Сортирует список группы по возрастанию алгоритмом "пузырька"

//Возвращает указатель на голову списка

Group\* SortGroups(Group\* Head)

{

Group\* X;

Group\* \_Head = Head;

int MinGroupNumber = Head->GroupNumebr;

//сортировка списка методом "пузырька"

for ( X = Head; X != nullptr; X = X->next)

{

for (Group\* Y = X->next; Y != nullptr; Y = Y->next)

{

if (X->GroupNumebr > Y->GroupNumebr)

{

swap(X->GroupNumebr, Y->GroupNumebr);

swap(X->FirstStudent, Y->FirstStudent);

}

if (X->GroupNumebr < MinGroupNumber)

{

MinGroupNumber = X->GroupNumebr;

\_Head = X;

}

}

}

Head = \_Head;

X = Head;

//обновление связей prev у отсортированного списка

X->prev = nullptr;

while (X->next != nullptr)

{

X->next->prev = X;

X = X->next;

}

return Head;

}

//Удаление студента

//Удаляет узел и списка студентов, обновляет связи после удаления

void DeleteStudent(Student\* x)

{

//Обновление связей между узлами

if ((x->prev == nullptr) && (x->next == nullptr))

{

}

else if (x->prev == nullptr)

{

(x->next)->prev = nullptr;

}

else if (x->next == nullptr)

{

(x->prev)->next = nullptr;

}

else

{

(x->prev)->next = x->next;

(x->next)->prev = x->prev;

}

delete x;

}

//Удаление группы

//Удаляет узел в списке групп. Обновляет связи после удаления

void DeleteGroup(Group\* X)

{

Student\* x = X->FirstStudent;

Student\* temp = x;

while (temp != nullptr)

{

temp = x->next;

DeleteStudent(x);

x = temp;

}

//Обновление связей между узлами

if ((X->prev == nullptr) && (X->next == nullptr))

{

}

else if (X->prev == nullptr)

{

(X->next)->prev = nullptr;

}

else if (X->next == nullptr)

{

(X->prev)->next = nullptr;

}

else

{

(X->prev)->next = X->next;

(X->next)->prev = X->prev;

}

delete temp;

delete X;

}

//Удаление списка групп

//Удаляет весь список групп

void DeleteAll(Group\* Head)

{

Group\* X = Head;

Group\* Temp = X;

while (Temp != nullptr)

{

Temp = X->next;

DeleteGroup(X);

X = Temp;

}

delete Temp;

}

//Печать шапки таблицы

//Выводит на экран шапку таблицы при выводе базы данных

void HeadOfPrint()

{

cout << "+--------------------------------------------------+--------------+------------------+---------------+" << endl;

cout << "| ФИО СТУДЕНТА | НОМЕР ГРУППЫ | ОЦЕНКИ ЗА СЕССИЮ | СТИПЕНДИЯ |" << endl;

cout << "+--------------------------------------------------+--------------+------------------+---------------+" << endl;

}

//Печать информации о студенте

//Выводит на консоль данные о текущем узле списка студентов

void PrintStudent(Student\* x, int \_groupNumber)

{

string marks;

for (int i = 0; i < 5; i++)

{

marks += to\_string(x->marks[i]);

marks += ',';

}

cout << "|" << setw(50) << x->name << "|" << setw(14) << \_groupNumber << "|" << setw(18) << marks

<< "|" << setw(15) << x->studentship << "|" << endl;

cout << "+--------------------------------------------------+--------------+------------------+---------------+" << endl;

}

//Печать информации

//Выводит данные обо всех узлах списка студентов группы, которые доступны по полю ->FirstStudent

void PrintGroup(Group\* X)

{

Student\* x;

x = X->FirstStudent;

while (x != nullptr)

{

PrintStudent(x, X->GroupNumebr);

x = x->next;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* Следующие функции служат для описания логики программы, ее архитектуру и логику

Они содержат в себе более простые функции, которые описаны выше

\*/

//Добавление студента

//Позволяет добавить студента в уже созданную пользователем группу.

Group\* AddStudent(Group\* X)

{

Group\* Head = X;

int \_tempGroupNumber;

cout << "Введите номер группы: ";

cin >> \_tempGroupNumber;

SearchGroup(X, \_tempGroupNumber); //Поиск группы для добавления

if (X== nullptr)

{

system("cls");

cout << "Такой группы не существует. Создайте новую группу в соответсвующем пункте меню. " << endl;

system("pause");

return Head;

}

else

{

system("cls");

cout << "Добавление студента в группу " << \_tempGroupNumber << "... " << endl;

Student\* x;

x = X->FirstStudent;

//находит пустой указатель для создания там нового узла

while (x->next != nullptr)

{

x = x->next;

}

//создает узел

x->next = CreateStudent(\_tempGroupNumber);

x->next->prev = x;

int action;

do

{

cout << "Студент добавлен в группу. Хотите ли вы добавить еще ?" << endl;

cout << "1 - да, любое другое значение - нет. ";

cin >> action;

if (action == 1)

{

system("cls");

x->next = CreateStudent(\_tempGroupNumber);

x->next->prev = x;

x = x->next;

}

} while (action == 1);

return Head;

}

}

//Добавить группу

//Позволяет добавить группу студентов

Group\* AddGroup(Group\* X)

{

Group\* Head;

Group\* \_prev = nullptr;

Head = X;

int \_tempGroupNumber;

cout << "Введите номер группы: ";

cin >> \_tempGroupNumber;

if (Head == nullptr)

{

Head = CreateGroup(\_tempGroupNumber);

return Head;

}

while (X != nullptr)

{

if (\_tempGroupNumber == X->GroupNumebr)

{

break;

}

\_prev = X;

X = X->next;

}

if (X != nullptr)

{

system("cls");

cout << "Такая группа уже была создана." << endl;

system("pause");

return Head;

}

else

{

X = new Group;

system("cls");

cout << "Создаем группу с номером " << \_tempGroupNumber << "... " << endl;

X = CreateGroup(\_tempGroupNumber);

X->prev = \_prev;

X->prev->next = X;

Head = SortGroups(Head);

return Head;

}

}

//Редактирование информации о студента

//Позволяет выборочно изменить данные о выбранном студенте

Group\* RedactInformationOfStudent(Group\* Head)

{

system("cls");

int \_tempGroupNumber;

cout << "Данная функция позволяет изменить информацию о студенте (ФИО, оценки за сессию и размер стипендии)" << endl;

//поиск студента

cout << "Введите номер группы: ";

cin >> \_tempGroupNumber;

Group\* X = SearchGroup(Head, \_tempGroupNumber);

if (X == nullptr)

{

cout << "Данные не найдены. " << endl;

system("pause");

return Head;

}

else

{

string FIO;

cout << "Введите ФИО студента: ";

std::cin.ignore(50, '\n');

getline(cin, FIO);

Student\* x = X->FirstStudent;

x = SearchStudent(x, FIO);

if (x == nullptr)

{

cout << "Данные не найдены. " << endl;

system("pause");

return Head;

}

else

{

//вывод на консоль данных о студенте, данные которого планируется редактировать

HeadOfPrint();

PrintStudent(x, X->GroupNumebr);

int action;

cout << "Хотите ли вы изменить ФИО студента ? " << endl;

cout << "1 - да, любая другая цифра - нет" << endl;

cout << "Выберите цифру для продолжения... ";

cin >> action;

if (action ==1 )

{

//изменение ФИО

x->name.clear();

do

{

cout << "Введите ФИО студента(не более 50 символов): ";

std::cin.ignore(50, '\n');

getline(cin, x->name);

if (x->name.size() > 50)

{

cout << "Количество символов в поле ФИО превышает допустимый лимит в 50 символов. Пожалуйста, введите данные еще раз." << endl;

x->name.clear();

}

} while ( x->name.empty());

}

cout << "Хотите ли вы изменить данные об оценках за последнюю сессию ?" << endl;

cout << "1 - да, любая другая цифра - нет" << endl;

cout << "Выберите цифру для продолжения... ";

cin >> action;

if (action == 1)

{

//изменение оценок

cout << "Введите последние пять оценок за сессию (без названий предметов)." << endl;

for (int i = 0; i < 5; i++)

{

x->marks[i] = 0;

}

int \_tempMark;

for (int i = 0; i < 5; i++)

{

while (x->marks[i] == 0)

{

cout << "Оценка " << i + 1 << ": ";

cin >> \_tempMark;

if (\_tempMark >= 2 && \_tempMark <= 5)

{

x->marks[i] = \_tempMark;

}

else

{

cout << "Оценка не может быть меньше двух или больше пяти. Введите данные еще раз: ";

}

}

}

}

cout << "Хотите ли вы изменить данные о стипендии студента ?" << endl;

cout << "1 - да, любая другая цифра - нет" << endl;

cout << "Выберите цифру для продолжения...";

cin >> action;

if (action == 1)

{

//изменение ступендии

cout << "Введите размер стипендии студента: ";

cin >> x->studentship;

}

//вывод новых данных о студенте

system("cls");

cout << "Новая информация о студенте " << x->name << ":" << endl;

HeadOfPrint();

PrintStudent(x, X->GroupNumebr);

system("pause");

X = Head;

return X;

}

}

}

//Сохранение информации о списке групп в файл

//Построчно сохраняет всю информацию о каждом студенте в текстовый файл .txt

//Между данными о студентах разных групп интервал пустая строка

void SaveToFile(string filename, Group\* Head)

{

ofstream fout;

fout.open(filename);

if (!fout.is\_open())

{

cout << "Не удалось открыть файл " << filename << " для записи. " << endl;

}

else

{

//сохраняет данные о студенте группы в строку через пробелы

Group\* X = Head;

Student\* x;

while (X != nullptr)

{

x = X->FirstStudent;

while (x != nullptr)

{

fout << x->name << " ";

fout << X->GroupNumebr << " ";

for (int i = 0; i < 5; i++)

{

fout << x->marks[i];

fout << " ";

}

fout << x->studentship;

fout << endl;

x = x->next;

}

//если данные о группе заканчиваются, отсупается одна строчка

X = X->next;

if (X != nullptr)

{

fout << endl;

}

}

string line;

cout << "Данные успешно сохранены В файл " << filename;

}

fout.close();

system("pause");

}

//Загрузка данных с файла

//Позволяет считать информацию о списке групп из текстового файла и заполнить ими список в программе

//Информация о считывается построчно, внутри строки - через пробелы с помощью указателя типа stringstream

//Если находит отступ в одну строку, перестает заполнять данные о группе и переходит к созданию следующей

//При окончании загрузки информации возвращает указатель на голову списка групп

Group\* AddinFile(string filename, Group\* Head)

{

ifstream fin;

fin.open(filename);

if (!fin.is\_open())

{

cout << "Не удалось открыть файл " << filename << " для чтения. " << endl;

}

else

{

if (Head != nullptr)

{

DeleteAll(Head);

}

Group\* X = new Group;

Head = nullptr;

X->prev = nullptr;

X->FirstStudent = nullptr;

string line;// Строчка из текстового файла

while (!fin.eof())//считывать файл до конца

{

if (Head == nullptr)

{

Head = X;

X->prev = nullptr;

}

else

{

X->next = new Group;

X->next->prev = X;

X = X->next;

}

Student\* x = new Student;

X->FirstStudent = nullptr;

x->prev = nullptr;

int j = 0;

while (getline(fin, line)) //считывать информацию из файла пока не конец строки

{

if (line == "")

{

break;

}

if (X->FirstStudent == nullptr)

{

X->FirstStudent = x;

x->prev = nullptr;

}

else

{

x->next = new Student;

x->next->prev = x;

x = x->next;

}

istringstream iss(line); //поток для чтения из строки

string token;//строка для хранения "слова" в строке

int i = 0;

//Читать строку line с помощью потока

//В зависимости от значения итератора сохранять данные из token в нужное инф.поле

while (getline(iss, token, ' ')) //

{

switch (i)

{

case 0: {

x->name = token;

break;

}

case 1: {

x->name += " " + token;

break;

}

case 2: {

x->name += " " + token;

break;

}

case 3:

{

if (j == 0)

{

X->GroupNumebr = atoi(token.c\_str());

j++;

}

break;

}

case 4: {

x->marks[i - 4] = atoi(token.c\_str());

break;

}

case 5: {

x->marks[i - 4] = atoi(token.c\_str());

break;

}

case 6: {

x->marks[i - 4] = atoi(token.c\_str());

break;

}

case 7: {

x->marks[i - 4] = atoi(token.c\_str());

break;

}

case 8: {

x->marks[i - 4] = atoi(token.c\_str());

break;

}

case 9: {

x->studentship = atoi(token.c\_str());

break;

}

default:

break;

}

i++;

}

}

x->next = nullptr;

}

X->next = nullptr;

fin.close();

cout << "Данные успешно загружены с файла. " << endl;

Head = SortGroups(Head);

}

system("pause");

return Head;

}

//Очистка потока от данных группы, которую выберет пользователь

void DeleteGroup\_MainMenuFunction(Group\* Head)

{

int \_tempGroupNumber;

cout << "Введите номер группы: ";

cin >> \_tempGroupNumber;

Group\* X = Head;

X = SearchGroup(X, \_tempGroupNumber);

if (X == nullptr)

{

cout << "Данные не найдены. " << endl;

}

else

{

if (X == Head)

{

if (X->next == nullptr)

{

cout << "Это единственная группа на потоке. Если хотите очистить данные - перейдите в соответствующий пункт главного меню." << endl;

system("pause");

return;

}

Head = X->next;

}

DeleteGroup(X);

cout << "Данные успешно удалены. " << endl;

}

system("pause");

}

//Очистка списка студентов группы от данных выбранного пользователем студента

void DeleteStudent\_MainMenuFunction(Group\* Head)

{

int \_tempGroupNumber;

cout << "Введите номер группы: ";

cin >> \_tempGroupNumber;

Group\* X = Head;

X = SearchGroup(X, \_tempGroupNumber);

if (X == nullptr)

{

cout << "Данные не найдены. " << endl;

}

else

{

string \_tempName;

cout << "Введите ФИО студента: ";

std::cin.ignore(50, '\n');

getline(cin, \_tempName);

Student\* x = X->FirstStudent;

x = SearchStudent(x, \_tempName);

if (x == nullptr)

{

cout << "Данные не найдены. " << endl;

}

else

{

if (x == X->FirstStudent)

{

if (x->next == nullptr)

{

cout << "В данной группе всего один студент. Если хотите удалить данную группу, перейдите в соответствующий пункт главного меню." << endl;

system("pause");

return;

}

X->FirstStudent = x->next;

}

DeleteStudent(x);

cout << "Данные успешно удалены. " << endl;

}

}

system("pause");

}

//Вывод данных студента на экран

//Выводит на экран данные о студенте при предварительном вводе ФИО для поиска

void PrintStudent\_MainMenuFunction(Group\* X)

{

int \_tempGroupNumber;

cout << "Введите номер группы студента: ";

cin >> \_tempGroupNumber;

X = SearchGroup(X, \_tempGroupNumber);

if (X == nullptr)

{

cout << "Даныне не найдены." << endl;

}

else

{

string \_TempName;

cout << "Введите ФИО студента: ";

std::cin.ignore(50, '\n');

getline(cin, \_TempName);

Student\* x = X->FirstStudent;

x = SearchStudent(x, \_TempName);

if (x == nullptr)

{

cout << "Данные не найдены." << endl;

}

else

{

HeadOfPrint();

PrintStudent(x, X->GroupNumebr);

}

}

system("pause");

}

//Вывод данных группы на экран

//Выводит на консоль данные о группе, найденной с помощью поиска по номеру группы

void PrintGroup\_MainMenuFunction(Group\* X)

{

int \_tempGroupNumber;

cout << "Введите номер группы: ";

cin >> \_tempGroupNumber;

X = SearchGroup(X, \_tempGroupNumber);

if (X == nullptr)

{

cout << "Данные не найдены. " << endl;

}

else

{

HeadOfPrint();

PrintGroup(X);

}

system("pause");

}

//Вывод данных потока

//Выводит на экран всю информацию о потоке

void PrintAll(Group\* X)

{

if (X == nullptr)

{

cout << "Данные не найдены." << endl;

}

else

{

while (X != nullptr)

{

HeadOfPrint();

PrintGroup(X);

X = X->next;

}

}

system("pause");

}

////////////////////////////////////////////////////////////////////////////////////////////////

/\*Функция для персонального задания\*/

void PersonalTuck(Group\* Head)

{

if (Head == nullptr)

{

cout << "Список групп пуст. Выполнение задания невозможно." << endl;

system("pause");

return;

}

cout << "Описание задания: " << endl;

cout << "Вывести в порядке возрастания номера групп, в которых соотношение " << endl;

cout << "суммарный объем стипендии/количество баллов превышает заданное значение." << endl << endl;

float DivVal;

cout << "Введите коэффициент отношения: ";

cin >> DivVal;

Group\* X = Head;

while (X != nullptr)

{

Student\* x = X->FirstStudent;

float DivEqual;

float SumStudentship = 0; // Суммарная стипендия

float SumBallsMarks = 0; //Сумма всех баллов, набранных студентами группы

//линейно обрабатывать список группы

//прибавлять данные о стипендии и оценках к созданным выше переменным

while (x != nullptr)

{

for (int i = 0; i < 5; i++)

{

SumBallsMarks += x->marks[i];

}

SumStudentship += x->studentship;

x = x->next;

}

//расчет получившегося значения

DivEqual = SumStudentship / SumBallsMarks;

//если оно получилось больше, чем то, которое мы ввели

//вывести номер группы на экран

if (DivEqual > DivVal)

{

cout << X->GroupNumebr << endl;

}

X = X->next;

}

cout << "У групп выше отношение суммарного объема стипендии" << endl;

cout << "больше заданного значения." << endl;

system("pause");

}

/////////////////////////////////////////////////////////////////////////////////////////////////

/\* Эта функция выводит на экран текст меню, а также позволяет обращаться к доступным

внутри него пользователю функциям.

\*/

void MainMenu(Group\* Head)

{

int action;

while (true)

{

system("cls");

cout << "ГЛАВНОЕ МЕНЮ " << endl;

cout << "1 - Добавить данные через консоль" << endl;

cout << "2 - Загрузить данные с файла" << endl;

cout << "3 - Вывести данные на экран" << endl;

cout << "4 - Сохранить данные в файл" << endl;

cout << "5 - Редактировать данные" << endl;

cout << "6 - Удалить данные" << endl;

cout << "7 - Демонстрация персонального задания" << endl;

cout << "0 - Выйти из программы" << endl << endl;

cout << "Выберите действие для продолжения: ";

cin >> action;

switch (action)

{

case 1: //Добавление данных через консоль

{

system("cls");

cout << "1 -Создать новую группу, 2 - добавить студента в уже созданную группу." << endl;

cout << "Выберите цифру для продолжения: ";

cin >> action;

system("cls");

switch (action)

{

case 1://Создание новой группы

{

Head = AddGroup(Head);

break;

}

case 2://Добавление нового студента в группу

{

Head = AddStudent(Head);

break;

}

default:

break;

}

break;

}

case 2://Загрузка данных с файла

{

Head = AddinFile("Potok.txt", Head);

break;

}

case 3://Вывод данных

{

system("cls");

cout << "1 -Вывести данные всех групп потока, 2 - Вывести данные группы, 3 - Вывести данные студента" << endl;

cout << "Выберите цифру для продолжения: ";

cin >> action;

system("cls");

switch (action)

{

case 1://Вывод потока

{

PrintAll(Head);

break;

}

case 2://Вывод группы

{

PrintGroup\_MainMenuFunction(Head);

break;

}

case 3://Вывод студента

{

PrintStudent\_MainMenuFunction(Head);

break;

}

default:

break;

}

break;

}

case 4://Сохранение данных в файл

{

SaveToFile("Potok.txt" , Head);

break;

}

case 5://Редактирование данных

{

Head = RedactInformationOfStudent(Head);

break;

}

case 6://Удаление данных

{

system("cls");

cout << "1 - Удалить данные всех групп потока, 2 - Удалить данные группы, 3 - Удалить данные студента" << endl;

cout << "Выберите цифру для продолжения: ";

cin >> action;

system("cls");

switch (action)

{

case 1://Удаление потока

{

DeleteAll(Head);

cout << "Данные успешно удалены." << endl;

Head = nullptr;

system("pause");

break;

}

case 2://Удаление группы

{

DeleteGroup\_MainMenuFunction(Head);

break;

}

case 3://Удаление студента

{

DeleteStudent\_MainMenuFunction(Head);

break;

}

default:

break;

}

break;

}

case 7: //Персональное задание

{

PersonalTuck(Head);

break;

}

case 0://Выход

{

DeleteAll(Head);

return;

}

default:

break;

}

}

}

int main()

{

system("chcp 1251");

Group\* Head = nullptr;

MainMenu(Head);

return 0;

}