**1. Описание алгоритма реализации операций над элементами списка**

**2. Описание программных элементов**

1. **Первый модуль. Прототипы функций обработки подсписка.**

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

// SPISOK.H contains main class interface and Item struct ///////////////////////////////////////////

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

#include "EXTRA FUNCTIONS.h"

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

// MAIN ITEM STRUCT TYPE ////////////////////////////////////////////////////////////////////////////

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

struct Stud

{

char\* ThirdName,\* FirstName,\* SecondName,\* JobType, \*Faculty;

int Salary;

Stud()

{

ThirdName=new char[80];

FirstName=new char[80];

SecondName=new char[80];

Faculty=new char[80];

JobType=new char[80];

}

};

struct Item

{

Stud info;

Item\* next;

Item\* prev;

};

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

// MAIN CLASS TYPE //////////////////////////////////////////////////////////////////////////////////

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

class Spisok

{

private:

// VARIABLES

Item \*Head,\*Student;

int CountStud;

FILE \*file, \*file1;

bool empty;

// METHODS

void StandardInput(int &,Item \*&); //Done! STANDARD INPUT EXTRA METHOD

void FoundMessage(int &,Item \*&); //Done! MESSAGE INDICATES SEARCH RESULTS

void SelectionMessage(); //Done! DISPLAYS FIELDS POSSIBLE FOR PROCESSING

public:

// METHODS

Spisok() //Done! CONSTRUCTOR

{

Head = new Item;

Head->next = NULL;

Student=Head;

CountStud=0;

}

~Spisok() //Done! DISTRUCTOR

{

delete Student;

delete Head;

}

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

//1) IO

void InputConsole(); //Done! INPUT FROM CONSOLE (REWRITE)

void OutputConsole(); //Done! INPUT FROM CONSOLE

void InputFile(char \*); //Done! INPUT FROM FILE (REWRITE)

void OutputFile(char \*); //Done! OUTPUT TO FILE

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

//2) DELETION

void DeleteElem(); // Done! DELETION OF ELEMENT FROM POSITION

void ViborFirst(); //Done! SELECTION OF FIRST ELEMENT (DELETE ALL EXCEPT FIRST) (CAR)

void DeleteFirst(); //Done! DELETE FIRST ELEMENT (CDR)

void ViborAndDeleteFirst(); //Done! CLEAN LIST (CADR)

void ResetAll(); //Done! RESET ALL LIST

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

//3) ADDITION

void DobavlElemVNach(); //Done! ADD ELEMENT TO HEAD

void DobavlElemVKonets(); //Done! ADD ELEMENT TO TAIL

void DobavlElemVPosiziy(); //Done! ADD ELEMENT TO POSITION

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

//4) CHANGE

void ChangeAllFields(); //Done! CHNGE ALL FIELDS OF STRUCTURE IN LIST

void ChangeVal(); //Done! CHNGE FIELD OF STRUCTURE IN LIST

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

//5) SPECIAL

void Search(); //Done! // SEARCH OF STUDENT IN LIST BY ANY KEY

void Reverse(); //Done! LIST REVERSE

void Sort(); //Done! // SORT BY ANY KEY

void Rakirovka(); //Done! TWO ELEMENTS ASSIGN EACH OTHER

void CopyTo(); //Done! COPY STUDENT TO OTHER POSITION

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

//6) OTHER

void Sound(); //Done! SOUND SIGNAL

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

//7) MAIN MENU FUNC

void MainMenu(char \*,char \*);

};

1. **Второй модуль. Реализация функций обработки подсписка.**

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

// SPISOK.CPP contains main class implementation ////////////////////////////////////////////////////

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

#include "SPISOK.h"

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

// PRIVATE EXTRA METHODS: ///////////////////////////////////////////////////////////////////////////

// STANDARD INPUT EXTRA METHOD

void Spisok::StandardInput(int &StudCounter,Item \*&List)

{

cout<<"СТУДЕНТ №"<<StudCounter<<":\n";

DrawLine();

cout<<"ВВЕДИТЕ ФАМИЛИЮ:\n";

cin>>List->info.ThirdName;

DrawLine();

cout<<"ВВЕДИТЕ ИМЯ:\n";

cin>>List->info.FirstName;

DrawLine();

cout<<"ВВЕДИТЕ ОТЧЕСТВО:\n";

cin>>List->info.SecondName;

DrawLine();

cout<<"ВВЕДИТЕ ФАКУЛЬТЕТ:\n";

cin>>List->info.Faculty;

DrawLine();

cout<<"ВВЕДИТЕ ТИП РАБОТ:\n";

cin>>List->info.JobType;

DrawLine();

cout<<"ВВЕДИТЕ ОКЛАД (ГРН):\n";

do

{

cin>>List->info.Salary;

InputErrorMsg();

}

while(List->info.Salary<1);

DrawLine();

}

// DISPLAYS FIELDS POSSIBLE FOR PROCESSING

void Spisok::SelectionMessage()

{

DrawLine();

cout<<"1) ФАМИЛИЯ\n";

DrawLine();

cout<<"2) ИМЯ\n";

DrawLine();

cout<<"3) ОТЧЕСТВО\n";

DrawLine();

cout<<"4) ФАКУЛЬТЕТ\n";

DrawLine();

cout<<"5) ТИП РАБОТ\n";

DrawLine();

cout<<"6) ОКЛАД\n";

DrawLine();

}

// MESSAGE INDICATES SEARCH RESULTS

void Spisok::FoundMessage(int & x,Item\* & buff)

{

cout<<"\nНАЙДЕНО В ПОЗИЦИИ: "<<x+1<<'\n';

cout<<"\nCТУДЕНТ №"<<x+1<<": ";

cout<<" "<<buff->info.ThirdName;

cout<<" "<<buff->info.FirstName;

cout<<" "<<buff->info.SecondName;

cout<<" "<<buff->info.Faculty;

cout<<" "<<buff->info.JobType;

cout<<" "<<buff->info.Salary;

cout<<endl;

DrawLine();

}

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

// PUBLIC MAIN METHODS: /////////////////////////////////////////////////////////////////////////////

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

// 1) INPUT-OUTPUT: /////////////////////////////////////////////////////////////////////////////////

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

// INPUT FROM CONSOLE

void Spisok::InputConsole()

{

do

{

clrscr;

cout<<"ВВЕДИТЕ КОЛИЧЕСТВО СТУДЕНТОВ <=15:\n";

cin>>CountStud;

InputErrorMsg();

}

while (CountStud<1|| CountStud>15);

Student=Head;

for (int i=1;i<CountStud+1;i++)

{

clrscr;

Student->next=new Item;

Student=Student->next;

StandardInput(i,Student);

Student->next=NULL;

}

}

// OUTPUT TO CONSOLE

void Spisok::OutputConsole()

{

Item \*buff;

buff=Head->next;

if (buff==NULL)

{

cout<<"ЗДЕСЬ ПОКА НЕТ НИ ОДНОЙ ЗАПИСИ...\n";

DrawLine();

empty=true;

}

else

{

empty=false;

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

{

cout<<"CТУДЕНТ №"<<i+1<<": ";

cout<<" "<<buff->info.ThirdName;

cout<<" "<<buff->info.FirstName;

cout<<" "<<buff->info.SecondName;

cout<<" "<<buff->info.Faculty;

cout<<" "<<buff->info.JobType;

cout<<" "<<buff->info.Salary;

cout<<endl;

buff=buff->next;

DrawLine();

}

}

}

// INPUT FROM FILE

void Spisok::InputFile(char \* s)

{

clrscr;

if ((file = fopen(s,"r")) == NULL)

{

cout<<"ОШИБКА ЧТЕНИЯ ИЗ ФАЙЛА!\n";

}

else

{

cout<<"ФАЙЛ ЗАГРУЖЕН!\n";

Student=Head;

CountStud=0;

while (!feof(file))

{

CountStud++;

char s[80];

Student->next=new Item;

Student=Student->next;

fscanf(file,"%s",Student->info.ThirdName);

fscanf(file,"%s",Student->info.FirstName);

fscanf(file,"%s",Student->info.SecondName);

fscanf(file,"%s",Student->info.Faculty);

fscanf(file,"%s",Student->info.JobType);

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

Student->info.Salary=atoi(s);

Student->next=NULL;

}

fclose(file);

DrawLine();

}

}

// OUTPUT TO FILE

void Spisok::OutputFile(char \*s)

{

clrscr;

if ((file = fopen(s,"w")) == NULL)

{

cout<<"ОШИБКА СОЗДАНИЯ ФАЙЛА!\n";

}

else

{

cout<<"ФАЙЛ СОХРАНЕН!\n";

Item \*buff;

buff=Head->next;

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

{

if (i) fprintf(file,"\n");

fprintf(file,"%s\n",buff->info.ThirdName);

fprintf(file,"%s\n",buff->info.FirstName);

fprintf(file,"%s\n",buff->info.SecondName);

fprintf(file,"%s\n",buff->info.Faculty);

fprintf(file,"%s\n",buff->info.JobType);

fprintf(file,"%d",buff->info.Salary);

buff=buff->next;

}

fclose(file);

DrawLine();

}

}

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

// 2) DELETION: /////////////////////////////////////////////////////////////////////////////////////

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

// DELETION OF ELEMENT FROM POSITION

void Spisok::DeleteElem()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

{

OutputConsole();

cout<<"\nВВЕДИТЕ НОМЕР СТУДЕНТА, КОТОРОГО ВЫ СОБИРАЕТЕСЬ УДАЛИТЬ ИЗ СПИСКА <="<<(CountStud?CountStud:1)<<":\n";

int InpPos;

do

{

cin>>InpPos;

InputErrorMsg();

}

while (InpPos<1 || InpPos>CountStud);

clrscr;

Item \*buff,\*buff2=new Item,\*buff3=new Item;

buff=Head;

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

buff=buff->next;

buff3=buff->next;

buff->next=NULL;

buff=Head;

for (int i=0;i<InpPos-1;i++)

buff=buff->next;

buff->next=new Item;

buff->next=buff3;

CountStud--;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

}

// SELECTION OF FIRST ELEMENT (DELETE ALL EXCEPT FIRST) (CAR)

void Spisok::ViborFirst()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

CountStud--;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

else

{

Item \*buff,\*bufhead;

buff=Head;

bufhead=buff;

buff=buff->next;

buff->next=NULL;

Student=bufhead;

CountStud=1;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

}

// DELETE FIRST ELEMENT (CDR)

void Spisok::DeleteFirst()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

{

Item \*buff,\*bufhead;

Student=Head;

buff=Head;

bufhead=buff;

buff->next=Student->next->next;

Student=bufhead;

CountStud--;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

}

// CLEAN LIST (CADR)

void Spisok::ViborAndDeleteFirst()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

{

ViborFirst();

DeleteFirst();

}

}

// RESET ALL LIST

void Spisok::ResetAll()

{

clrscr;

Student=Head;

Student->next=NULL;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

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

// 3) ADDITION: /////////////////////////////////////////////////////////////////////////////////////

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

// ADD ELEMENT TO HEAD

void Spisok::DobavlElemVNach()

{

clrscr;

Item \*buff,\*bufhead;

bufhead=new Item;

buff=bufhead;

buff->next=new Item;

buff=buff->next;

int i=1;

StandardInput(i,buff);

buff->next=NULL;

Student=Head->next;

buff->next=new Item;

buff->next=Student;

Head=bufhead;

Student=Head;

CountStud++;

}

// ADD ELEMENT TO TAIL

void Spisok::DobavlElemVKonets()

{

clrscr;

Student->next=new Item;

Student=Student->next;

Student->next=NULL;

CountStud++;

StandardInput(CountStud,Student);

}

// ADD ELEMENT TO POSITION

void Spisok::DobavlElemVPosiziy()

{

clrscr;

cout<<"ВВЕДИТЕ ПОЗИЦИЮ В СПИСКЕ, В КОТОРУЮ СОБИРАЕТЕСЬ ДОБАВИТЬ СТУДЕНТА <="<<(CountStud?CountStud:1)<<":\n";

int InpPos;

do

{

cin>>InpPos;

InputErrorMsg();

}

while (InpPos<1 || InpPos>CountStud);

clrscr;

Item \*buff,\*buff2=new Item,\*buff3=new Item;

buff=Head;

for (int i=0;i<InpPos-1;i++)

buff=buff->next;

buff3=buff->next;

buff->next=NULL;

StandardInput(InpPos,buff2);

buff->next=new Item;

buff->next=buff2;

buff=buff->next;

buff->next=new Item;

buff->next=buff3;

CountStud++;

}

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

// 4) CHANGE: ///////////////////////////////////////////////////////////////////////////////////////

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

// CHNGE ALL FIELDS OF STRUCTURE IN LIST

void Spisok::ChangeAllFields()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

{

clrscr;

OutputConsole();

cout<<"\nВВЕДИТЕ НОМЕР СТУДЕНТА, ДАННЫЕ О КОТОРОМ ХОТИТЕ ИЗМЕНИТЬ: <="<<CountStud<<'\n';

int NumStud;

do

{

cin>>NumStud;

InputErrorMsg();

}

while (NumStud<1 || NumStud>CountStud);

clrscr;

Item \*buff;

buff=Head;

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

buff=buff->next;

StandardInput(NumStud,buff);

}

}

// CHNGE FIELD OF STRUCTURE IN LIST

void Spisok::ChangeVal()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

{

clrscr;

OutputConsole();

cout<<"\nВВЕДИТЕ НОМЕР СТУДЕНТА, ДАННЫЕ О КОТОРОМ ХОТИТЕ ИЗМЕНИТЬ: <="<<CountStud<<'\n';

DrawLine();

int NumStud;

do

{

cin>>NumStud;

InputErrorMsg();

}

while (NumStud<1 || NumStud>CountStud);

clrscr;

Item \*buff;

buff=Head;

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

buff=buff->next;

cout<<"ВВЕДИТЕ ПОЛЕ, КОТОРОЕ ХОТИТЕ ИЗМЕНИТЬ: <="<<CountStud<<'\n';

SelectionMessage();

int FieldNum;

cin>>FieldNum;

switch(FieldNum)

{

case 1:

{

cout<<"ВВЕДИТЕ ФАМИЛИЮ:\n";

cin>>buff->info.ThirdName;

break;

}

case 2:

{

cout<<"ВВЕДИТЕ ИМЯ:\n";

cin>>buff->info.FirstName;

break;

}

case 3:

{

cout<<"ВВЕДИТЕ ОТЧЕСТВО:\n";

cin>>buff->info.SecondName;

break;

}

case 4:

{

cout<<"ВВЕДИТЕ ФАКУЛЬТЕТ:\n";

cin>>buff->info.Faculty;

break;

}

case 5:

{

cout<<"ВВЕДИТЕ ТИП РАБОТ:\n";

cin>>buff->info.JobType;

break;

}

case 6:

{

cout<<"ВВЕДИТЕ ОКЛАД (ГРН):\n";

do

{

cin>>buff->info.Salary;

InputErrorMsg();

}

while (buff->info.Salary<1);

break;

}

}

}

}

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

// 5) SPECIAL: //////////////////////////////////////////////////////////////////////////////////////

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

// SEARCH OF STUDENT IN LIST BY ANY KEY

void Spisok::Search()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

if (CountStud<2)

{

{

cout<<"СПИСОК СЛИШКОМ КОРОТКИЙ!\n";

DrawLine();

}

}

else

{

cout<<"ВЫБЕРИТЕ ПОЛЕ, ПО КОТОРОМУ ВЫ ХОТИТЕ ПРОИЗВЕСТИ ПОИСК: <="<<CountStud<<'\n';

SelectionMessage();

cout<<"7) НОМЕР СТУДЕНТА\n";

DrawLine();

int FieldNum;

do

{

cin>>FieldNum;

InputErrorMsg();

}

while (FieldNum<1 || FieldNum >7);

int n;

char \*s=new char[80];

clrscr;

OutputConsole();

Item \*buff;

buff=Head->next;

int i,x;

bool found;

switch(FieldNum)

{

case 1:

{

found=false;

cout<<"ВВЕДИТЕ ФАМИЛИЮ:\n";

cin>>s;

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (!strcmp(buff->info.ThirdName,s))

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

case 2:

{

found=false;

cout<<"ВВЕДИТЕ ИМЯ:\n";

cin>>s;

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (!strcmp(buff->info.FirstName,s))

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

case 3:

{

found=false;

cout<<"ВВЕДИТЕ ОТЧЕСТВО:\n";

cin>>s;

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (!strcmp(buff->info.SecondName,s))

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

case 4:

{

found=false;

cout<<"ВВЕДИТЕ ФАКУЛЬТЕТ:\n";

cin>>s;

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (!strcmp(buff->info.Faculty,s))

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

case 5:

{

found=false;

cout<<"ВВЕДИТЕ ТИП РАБОТ:\n";

cin>>s;

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (!strcmp(buff->info.JobType,s))

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

case 6:

{

found=false;

cout<<"ВВЕДИТЕ ОКЛАД (ГРН):\n";

do

{

cin>>n;

InputErrorMsg();

}

while(n<1);

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (buff->info.Salary==n)

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

case 7:

{

found=false;

cout<<"ВВЕДИТЕ НОМЕР СТУДЕНТА:\n";

do

{

cin>>n;

InputErrorMsg();

}

while (n>CountStud || n<1);

OutputConsole();

for (i=0;i<CountStud;i++)

{

if (i+1==n)

{

found=true;

FoundMessage(i,buff);

}

buff=buff->next;

}

x=i+1;

break;

}

}

if (!found)

{

clrscr;

cout<<"СТУДЕНТ НЕ НАЙДЕН!\n";

DrawLine();

}

}

}

// LIST REVERSE

void Spisok::Reverse()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

if (CountStud<2)

{

{

cout<<"СПИСОК СЛИШКОМ КОРОТКИЙ!\n";

DrawLine();

}

}

else

{

Item \*buff=Head->next,\*buff2,\*buff2head;

buff2head=new Item;

buff2head->next=NULL;

buff2=buff2head;

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

{

buff=Head->next;

for (int i=0;i<CountStud-j-1;i++)

buff=buff->next;

buff2->next=new Item;

buff2=buff2->next;

buff2->info=buff->info;

buff2->next=NULL;

}

buff2=buff2head->next;

buff=buff2;

Head=buff2head;

cout<<"СПИСОК ОБРАЩЕН!\n";

DrawLine();

}

}

// SORT BY ANY KEY

void Spisok::Sort()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

if (CountStud<2)

{

{

cout<<"СПИСОК СЛИШКОМ КОРОТКИЙ!\n";

DrawLine();

}

}

else

{

cout<<"ВЫБЕРИТЕ ПОЛЕ, ПО КОТОРОМУ ВЫ ХОТИТЕ ПРОИЗВЕСТИ СОРТИРОВКУ: <="<<CountStud<<'\n';

SelectionMessage();

int FieldNum;

do

{

cin>>FieldNum;

InputErrorMsg();

}

while (FieldNum<1 || FieldNum >6);

Item \*trash = new Item,\*buff=Head;

switch(FieldNum)

{

case 1:

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

{

buff=Head->next;

for (int j=0; j < CountStud-i-1; j++)

{

if (strcmp(buff->info.ThirdName,buff->next->info.ThirdName)>0)

{

trash->info=buff->info;

buff->info=buff->next->info;

buff->next->info=trash->info;

}

buff=buff->next;

}

}

Student=buff;

break;

case 2:

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

{

buff=Head->next;

for (int j=0; j < CountStud-i-1; j++)

{

if (strcmp(buff->info.FirstName,buff->next->info.FirstName)>0)

{

trash->info=buff->info;

buff->info=buff->next->info;

buff->next->info=trash->info;

}

buff=buff->next;

}

}

Student=buff;

break;

case 3:

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

{

buff=Head->next;

for (int j=0; j < CountStud-i-1; j++)

{

if (strcmp(buff->info.SecondName,buff->next->info.SecondName)>0)

{

trash->info=buff->info;

buff->info=buff->next->info;

buff->next->info=trash->info;

}

buff=buff->next;

}

}

Student=buff;

break;

case 4:

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

{

buff=Head->next;

for (int j=0; j < CountStud-i-1; j++)

{

if (strcmp(buff->info.Faculty,buff->next->info.Faculty)>0)

{

trash->info=buff->info;

buff->info=buff->next->info;

buff->next->info=trash->info;

}

buff=buff->next;

}

}

Student=buff;

break;

case 5:

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

{

buff=Head->next;

for (int j=0; j < CountStud-i-1; j++)

{

if (strcmp(buff->info.JobType,buff->next->info.JobType)>0)

{

trash->info=buff->info;

buff->info=buff->next->info;

buff->next->info=trash->info;

}

buff=buff->next;

}

}

Student=buff;

break;

case 6:

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

{

buff=Head->next;

for (int j=0; j < CountStud-i-1; j++)

{

if (buff->info.Salary>buff->next->info.Salary)

{

trash->info=buff->info;

buff->info=buff->next->info;

buff->next->info=trash->info;

}

buff=buff->next;

}

}

Student=buff;

break;

}

clrscr;

cout<<"ИНФОРМАЦИЯ ОТСОРТИРОВАНА!\n";

DrawLine();

}

}

// TWO ELEMENTS ASSIGN EACH OTHER

void Spisok::Rakirovka()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

if (CountStud<2)

{

{

cout<<"СПИСОК СЛИШКОМ КОРОТКИЙ!\n";

DrawLine();

}

}

else

{

clrscr;

OutputConsole();

cout<<"ВВЕДИТЕ ПОЗИЦИЮ ПЕРВОГО СТУДЕНТА В СПИСКЕ <="<<(CountStud?CountStud:1)<<":\n";

int InpPos1;

do

{

cin>>InpPos1;

InputErrorMsg();

}

while (InpPos1>CountStud ||InpPos1<1);

cout<<"ВВЕДИТЕ ПОЗИЦИЮ, В КОТОРУЮ СОБИРАЕТЕСЬ ВСТАВИТЬ СТУДЕНТА <="<<(CountStud?CountStud:1)<<":\n";

int InpPos2;

do

{

cin>>InpPos2;

InputErrorMsg();

}

while (InpPos2==InpPos1 || InpPos2>CountStud || InpPos2<1);

Item \*buff,\*buff1=new Item,\*buff2,\*buff3=new Item;

buff=Head->next;

for(int i=0;i<InpPos1-1;i++)

{

buff=buff->next;

}

buff1=buff;

buff=Head->next;

for(int i=0;i<InpPos2-1;i++)

{

buff=buff->next;

}

buff2=buff;

buff3->info=buff1->info;

buff1->info=buff2->info;

buff2->info=buff3->info;

buff=Head;

clrscr;

cout<<"ИНФОРМАЦИЯ ИЗМЕНЕНА!\n";

DrawLine();

}

}

// COPY STUDENT TO OTHER POSITION

void Spisok::CopyTo()

{

clrscr;

if (empty)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

if (CountStud<2)

{

{

cout<<"СПИСОК СЛИШКОМ КОРОТКИЙ!\n";

DrawLine();

}

}

else

{

clrscr;

OutputConsole();

cout<<"ВВЕДИТЕ ПОЗИЦИЮ В СПИСКЕ СТУДЕНТА, КОТОРОГО ВЫ СОБИРАЕТЕСЬ КОПИРОВАТЬ <="<<(CountStud?CountStud:1)<<":\n";

int InpPos1;

do

{

cin>>InpPos1;

InputErrorMsg();

}

while (InpPos1>CountStud ||InpPos1<1);

cout<<"ВВЕДИТЕ ПОЗИЦИЮ, В КОТОРУЮ СОБИРАЕТЕСЬ ВСТАВИТЬ СТУДЕНТА <="<<(CountStud?CountStud:1)<<":\n";

int InpPos2;

do

{

cin>>InpPos2;

InputErrorMsg();

}

while (InpPos2==InpPos1 || InpPos2>CountStud || InpPos2<1);

Item \*buff,\*buff1=new Item,\*buff2,\*buff3=new Item;

buff=Head->next;

for(int i=0;i<InpPos1-1;i++)

{

buff=buff->next;

}

buff1=buff;

buff=Head->next;

for(int i=0;i<InpPos2-1;i++)

{

buff=buff->next;

}

buff2=buff;

buff2->info=buff1->info;

buff=Head;

clrscr;

cout<<"ИНФОРМАЦИЯ ИЗМЕНЕНА!\n";

DrawLine();

}

}

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

// 6) OTHER: ////////////////////////////////////////////////////////////////////////////////////////

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

// SOUND SIGNAL

void Spisok::Sound()

{

Beep(1000,200);

}

void Spisok::MainMenu(char \*path,char \*pathOut)

{

bool end=false;

char Key;

do

{

clrscr;

setlocale(0,"Rus");

cout<<"STUDENTS DATABASE | БАЗА ДАННЫХ О СТУДЕНТАХ\n";

DrawLine();

cout<<"1) ВВЕСТИ ИНФОРМАЦИЮ С КЛАВИАТУРЫ (С ПЕРЕЗАПИСЬЮ ТЕКУЩЕГО СПИСКА)\n";

DrawLine();

cout<<"2) ПОКАЗАТЬ ТЕКУЩУЮ ИНФОРМАЦИЮ\n";

DrawLine();

cout<<"3) ОТКРЫТЬ ФАЙЛ (С ПЕРЕЗАПИСЬЮ ТЕКУЩЕГО СПИСКА)\n";

DrawLine();

cout<<"4) СОХРАНИТЬ ТЕКУЩУЮ ИНФОРМАЦИЮ В ФАЙЛ\n";

DrawLine();

cout<<"5) ОПЕРАЦИИ НАД СПИСКОМ\n";

DrawLine();

cout<<"6) УДАЛИТЬ ТЕКУЩУЮ ИНФОРМАЦИЮ\n";

DrawLine();

cout<<"7) ИЗМЕНИТЬ ИНФОРМАЦИОННЫЕ ПОЛЯ\n";

DrawLine();

cout<<"8) ЗВУКОВОЙ СИГНАЛ\n";

DrawLine();

cout<<"ESC - ВЫХОД\n";

DrawLine();

Key=\_getch();

switch(Key)

{

case '1':InputConsole(); WaitForPress();break;

case '2':clrscr;OutputConsole(); WaitForPress();break;

case '3':InputFile(path); WaitForPress();break;

case '4':OutputFile(pathOut); WaitForPress();break;

case '5':

clrscr;

cout<<"1) СОРТИРОВАТЬ ИНФОРМАЦИЮ\n";

DrawLine();

cout<<"2) ПОИСК СТУДЕНТА\n";

DrawLine();

cout<<"3) ДОБАВИТЬ СТУДЕНТА\n";

DrawLine();

cout<<"4) РЕВЕРС СПИСКА СТУДЕНТОВ\n";

DrawLine();

cout<<"5) ПОМЕНЯТЬ МЕСТАМИ СТУДЕНТОВ В СПИСКЕ\n";

DrawLine();

cout<<"6) КОПИРОВАТЬ СТУДЕНТА В ПОЗИЦИЮ\n";

DrawLine();

cout<<"ESC - ВОЗВРАТ\n";

DrawLine();

do

Key=\_getch();

while (Key!='1' && Key!='2' && Key!='3' && Key!='4' && Key!='5' && Key!='6' && Key!=27);

switch(Key)

{

case '1':Sort();WaitForPress();break;

case '2':Search();WaitForPress();break;

case '3':

clrscr;

cout<<"1) ДОБАВИТЬ В КОНЕЦ СПИСКА\n";

DrawLine();

cout<<"2) ДОБАВИТЬ В НАЧАЛО СПИСКА\n";

DrawLine();

cout<<"3) ДОБАВИТЬ В ПОЗИЦИЮ\n";

DrawLine();

cout<<"ESC - ВОЗВРАТ\n";

DrawLine();

Key=\_getch();

switch(Key)

{

case '1':DobavlElemVKonets(); WaitForPress();break;

case '2':DobavlElemVNach(); WaitForPress();break;

case '3':DobavlElemVPosiziy(); WaitForPress();break;

}

break;

case '4':Reverse();WaitForPress();break;

case '5':Rakirovka();WaitForPress();break;

case '6':CopyTo();WaitForPress();break;

}

break;

case '6':

clrscr;

cout<<"1) УДАЛИТЬ ИНФОРМАЦИЮ О КОНКРЕТНОМ СТУДЕНТЕ\n";

DrawLine();

cout<<"2) УДАЛИТЬ ИНФОРМАЦИЮ О ВСЕХ КРОМЕ ПЕРВОГО (CAR)\n";

DrawLine();

cout<<"3) УДАЛИТЬ ИНФОРМАЦИЮ О ПЕРВОМ СТУДЕНТЕ (CDR)\n";

DrawLine();

cout<<"4) УДАЛИТЬ ВСЮ ИНФОРМАЦИЮ (CADR)\n";

DrawLine();

cout<<"5) УДАЛИТЬ ВСЮ ИНФОРМАЦИЮ (ОСВОБОЖДЕНИЕ ПАМЯТИ)\n";

DrawLine();

cout<<"ESC - ВОЗВРАТ\n";

DrawLine();

do

Key=\_getch();

while (Key!='1' && Key!='2' && Key!='3' && Key!='4' &&Key!='5' && Key!=27);

switch(Key)

{

case '1':DeleteElem(); WaitForPress();break;

case '2':ViborFirst(); WaitForPress();break;

case '3':DeleteFirst(); WaitForPress();break;

case '4':ViborAndDeleteFirst(); WaitForPress();break;

case '5':ResetAll(); WaitForPress();break;

case 27:break;

}

break;

case '7':

clrscr;

cout<<"1) ИЗМЕНИТЬ ВСЕ ПОЛЯ\n";

DrawLine();

cout<<"2) ИЗМЕНИТЬ ОДНО ПОЛЕ\n";

DrawLine();

cout<<"ESC - ВОЗВРАТ\n";

DrawLine();

do

Key=\_getch();

while (Key!='1' && Key!='2' && Key!=27);

switch(Key)

{

case '1':ChangeAllFields();WaitForPress();break;

case '2':ChangeVal();WaitForPress();break;

case 27:break;

}

break;

case '8':Sound();break;

case 27:end=true;

}

}

while (!end);

}

1. **Третий модуль. Прототипы функций обработки разветвленного списка.**

#include "SPISOK.h"

struct ItemMult

{

char \*UniversityName;

Spisok ListCS;

ItemMult \*next;

ItemMult \*prev;

ItemMult()

{

UniversityName=new char[80];

}

};

class MultiSpisok

{

private:

// VARIABLES

ItemMult \*Head, \* University;

int CountUniver;

bool emptyMult;

public:

// METHODS

MultiSpisok()

{

Head = new ItemMult;

Head->next = NULL;

University=Head;

University=0;

}

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

//1) IO

void InputConsole(); //Done! INPUT FROM CONSOLE (REWRITE)

void OutputConsole(); //Done! INPUT FROM CONSOLE

void InputFile(); //Done! INPUT FROM FILE (REWRITE)

void OutputFile(); //Done! OUTPUT TO FILE

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

//3) SELECTION

void SelectUniversity();

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

//2) DELETION

void DeleteElem(); // Done! DELETION OF ELEMENT FROM POSITION

void ResetAll(); //Done! RESET ALL LIST

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

//3) ADDITION

void DobavlElemVPosiziy(); //Done! ADD ELEMENT TO POSITION

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

//4) SPECIAL

void Search(); //Done! // SEARCH OF STUDENT IN LIST BY ANY KEY

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

//5) EXTRA

void Sound();

};

1. **Четвертый модуль. Реализация функций разветвленного списка.**

#include "MULTI SPISOK.h"

void MultiSpisok::InputConsole()

{

do

{

clrscr;

cout<<"ВВЕДИТЕ КОЛИЧЕСТВО УНИВЕРСИТЕТОВ <=15:\n";

cin>>CountUniver;

InputErrorMsg();

}

while (CountUniver<1|| CountUniver>15);

University=Head;

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

{

clrscr;

University->next=new ItemMult;

University=University->next;

cout<<"ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА №"<<i+1<<":\n";

cin>>University->UniversityName;

University->next=NULL;

}

}

void MultiSpisok::OutputConsole()

{

clrscr;

ItemMult \*buff;

buff=Head->next;

if (buff==NULL)

{

cout<<"СПИСОК УНИВЕРСИТЕТОВ ПУСТ...\n";

DrawLine();

emptyMult=true;

}

else

{

emptyMult=false;

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

{

cout<<i+1<<") "<<buff->UniversityName<<": \n";

buff->ListCS.OutputConsole();

buff=buff->next;

}

}

}

void MultiSpisok::InputFile()

{

University=Head;

emptyMult=false;

University->next=new ItemMult;

University=University->next;

char \*s1="DonNU";

char \*s2="DonNTU";

University->UniversityName=s1;

University->ListCS.InputFile("DonNU.tdt");

University->next=NULL;

University->next=new ItemMult;

University=University->next;

University->UniversityName=s2;

University->ListCS.InputFile("DonNTU.tdt");

University->next=NULL;

CountUniver=2;

}

void MultiSpisok::OutputFile()

{

clrscr;

ItemMult \*buff;

buff=Head->next;

if (buff==NULL)

{

cout<<"СПИСОК УНИВЕРСИТЕТОВ ПУСТ...\n";

DrawLine();

emptyMult=true;

}

else

{

char \*s=new char[80];

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

{

sprintf(s,"%s%s",buff->UniversityName,"Out.tdt");

buff->ListCS.OutputFile(s);

buff=buff->next;

}

}

}

void MultiSpisok::SelectUniversity()

{

clrscr;

ItemMult \*buff,\*bufhead=NULL;

buff=Head->next;

if (buff==NULL)

{

cout<<"СПИСОК УНИВЕРСИТЕТОВ ПУСТ...\n";

DrawLine();

WaitForPress();

emptyMult=true;

}

else

{

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

{

cout<<i+1<<") "<<buff->UniversityName<<"\n";

buff=buff->next;

}

buff=Head;

cout<<"ВЫБЕРИТЕ УНИВЕРСИТЕТ, ИНФОРМАЦИЮ О СТУДЕНТАХ КОТОРОГО ХОТИТЕ ИЗМЕНИТЬ <"<<CountUniver<<":\n";

int NumUniver;

cin>> NumUniver;

University=Head;

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

University=University->next;

char s[80],s1[80];

sprintf(s,"%s%s",University->UniversityName,".tdt");

sprintf(s1,"%s%s",University->UniversityName,"Out.tdt");

University->ListCS.MainMenu(s,s1);

}

}

void MultiSpisok::DeleteElem()

{

clrscr;

if (emptyMult)

{

cout<<"СПИСОК ПУСТ!\n";

DrawLine();

}

else

{

ItemMult \*buff,\*buff2=new ItemMult,\*buff3=new ItemMult;

buff=Head->next;

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

{

cout<<i+1<<") "<<buff->UniversityName<<"\n";

buff=buff->next;

}

buff=Head;

cout<<"\nВВЕДИТЕ НОМЕР УНИВЕРСИТЕТА, КОТОРОГО ВЫ СОБИРАЕТЕСЬ УДАЛИТЬ ИЗ СПИСКА <="<<(CountUniver?CountUniver:1)<<":\n";

int InpPos;

do

{

cin>>InpPos;

InputErrorMsg();

}

while (InpPos<1 || InpPos>CountUniver);

clrscr;

buff=Head;

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

buff=buff->next;

buff3=buff->next;

buff->next=NULL;

buff=Head;

for (int i=0;i<InpPos-1;i++)

buff=buff->next;

buff->next=new ItemMult;

buff->next=buff3;

CountUniver--;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

}

void MultiSpisok::ResetAll()

{

clrscr;

University=Head;

University->next=NULL;

cout<<"ИНФОРМАЦИЯ УДАЛЕНА!\n";

DrawLine();

}

void MultiSpisok::DobavlElemVPosiziy()

{

clrscr;

cout<<"ВВЕДИТЕ ПОЗИЦИЮ В СПИСКЕ, В КОТОРУЮ СОБИРАЕТЕСЬ ДОБАВИТЬ СТУДЕНТА <="<<(CountUniver?CountUniver:1)<<":\n";

int InpPos;

do

{

cin>>InpPos;

InputErrorMsg();

}

while (InpPos<1 || InpPos>CountUniver);

clrscr;

ItemMult \*buff,\*buff2=new ItemMult,\*buff3=new ItemMult;

buff=Head;

for (int i=0;i<InpPos-1;i++)

buff=buff->next;

buff3=buff->next;

buff->next=NULL;

cout<<"ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА №"<<InpPos<<":\n";

cin>>buff2->UniversityName;

buff->next=new ItemMult;

buff->next=buff2;

buff=buff->next;

buff->next=new ItemMult;

buff->next=buff3;

CountUniver++;

}

void MultiSpisok::Search()

{

ItemMult \*buff;

buff=Head->next;

int i,x;

bool found;

found=false;

char \*s=new char[80];

clrscr;

cout<<"ВЫБЕРИТЕ ПОЛЕ, ПО КОТОРОМУ ВЫ ХОТИТЕ ПРОИЗВЕСТИ ПОИСК: <="<<CountUniver<<'\n';

DrawLine();

cout<<"1) ПО НАЗВАНИЮ УНИВЕРСИТЕТА\n";

DrawLine();

cout<<"2) ПО НОМЕРУ УНИВЕРСИТЕТА\n";

DrawLine();

int FieldNum;

do

{

cin>>FieldNum;

InputErrorMsg();

}

while (FieldNum<1 || FieldNum >2);

clrscr;

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

{

cout<<i+1<<") "<<buff->UniversityName<<"\n";

buff=buff->next;

}

buff=Head->next;

if (FieldNum==1)

{

cout<<"ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА:\n";

cin>>s;

for (i=0;i<CountUniver;i++)

{

if (!strcmp(buff->UniversityName,s))

{

found=true;

cout<<"\nНАЙДЕНО В ПОЗИЦИИ: "<<i+1<<'\n';

cout<<i+1<<") "<<buff->UniversityName<<"\n";

}

buff=buff->next;

}

}

else

{

cout<<"ВВЕДИТЕ НОМЕР УНИВЕРСИТЕТА:\n";

cin>>x;

for (i=0;i<CountUniver;i++)

{

if (i+1==x)

{

found=true;

cout<<"\nНАЙДЕНО В ПОЗИЦИИ: "<<i+1<<'\n';

cout<<i+1<<") "<<buff->UniversityName<<"\n";

}

buff=buff->next;

}

}

}

void MultiSpisok::Sound()

{

ItemMult \*buff;

buff=Head;

buff->ListCS.Sound();

}

1. **Пятый модуль. Реализация функций для работы с меню.**

#include "Main.h"

#include "MULTI SPISOK.h"

void MainMenu(MultiSpisok &); // OVERLOAD MAIN FUNCTION

void main()

{

SetColor(Blue,White);

MultiSpisok ListCS;

MainMenu(ListCS);

}

void MainMenu(MultiSpisok & MultiListCS) // OVERLOAD MAIN FUNCTION

{

char Key1;

do

{

clrscr;

setlocale(0,"Rus");

cout<<"STUDENTS DATABASE | БАЗА ДАННЫХ О СТУДЕНТАХ УНИВЕРСИТЕТОВ ДОНЕЦКА\n";

DrawLine();

cout<<"1) ПЕРЕЙТИ К ВЫБОРУ УНИВЕРСИТЕТОВ\n";

DrawLine();

cout<<"2) ВВЕСТИ ИНФОРМАЦИЮ С КЛАВИАТУРЫ (С ПЕРЕЗАПИСЬЮ ТЕКУЩЕГО СПИСКА)\n";

DrawLine();

cout<<"3) ПОКАЗАТЬ СПИСОК УНИВЕРСИТЕТОВ\n";

DrawLine();

cout<<"4) ОТКРЫТЬ ФАЙЛ (С ПЕРЕЗАПИСЬЮ ТЕКУЩЕГО СПИСКА УНИВЕРСИТЕТОВ)\n";

DrawLine();

cout<<"5) СОХРАНИТЬ СПИСОК УНИВЕРСИТЕТОВ\n";

DrawLine();

cout<<"6) ОПЕРАЦИИ НАД СПИСКОМ УНИВЕРСИТЕТОВ\n";

DrawLine();

cout<<"7) УДАЛИТЬ ТЕКУЩУЮ ИНФОРМАЦИЮ\n";

DrawLine();

cout<<"8) ЗВУКОВОЙ СИГНАЛ\n";

DrawLine();

cout<<"ESC - ВЫХОД\n";

Key1=\_getch();

switch(Key1)

{

case '1':MultiListCS.SelectUniversity(); break;

case '2':MultiListCS.InputConsole(); WaitForPress();break;

case '3':MultiListCS.OutputConsole(); WaitForPress();break;

case '4':MultiListCS.InputFile(); WaitForPress();break;

case '5':MultiListCS.OutputFile(); WaitForPress();break;

case '6':

clrscr;

cout<<"1) ПОИСК СТУДЕНТА\n";

DrawLine();

cout<<"2) ДОБАВИТЬ УНИВЕРСИТЕТ В ПОЗИЦИЮ\n";

DrawLine();

cout<<"ESC - ВОЗВРАТ\n";

DrawLine();

do

Key1=\_getch();

while (Key1!='1' && Key1!='2'&& Key1!=27);

switch (Key1)

{

case '1':MultiListCS.Search(); WaitForPress();break;

case '2':MultiListCS.DobavlElemVPosiziy(); WaitForPress();break;

}

break;

case '7':

clrscr;

cout<<"1) УДАЛИТЬ ИНФОРМАЦИЮ О КОНКРЕТНОМ УНИВЕРСИТЕТЕ\n";

DrawLine();

cout<<"2) УДАЛИТЬ ВСЮ ИНФОРМАЦИЮ (ОСВОБОЖДЕНИЕ ПАМЯТИ)\n";

DrawLine();

cout<<"ESC - ВОЗВРАТ\n";

DrawLine();

do

Key1=\_getch();

while (Key1!='1' && Key1!='2'&& Key1!=27);

switch (Key1)

{

case '1':MultiListCS.DeleteElem(); WaitForPress();break;

case '2':MultiListCS.ResetAll(); WaitForPress();break;

}

case '8':MultiListCS.Sound();break;

case 27:exit(0);

}

}

while (1);

}

1. **Шестой модуль. Прототипы вспомогательных функций .**

#include <iostream>

#include <conio.h>

#include <Windows.h>

#define clrscr system("cls");

using namespace std;

enum ConsoleColor // 16 BIT COLOR TYPE DEFINITION

{

Black = 0,

Blue = 1,

Green = 2,

Cyan = 3,

Red = 4,

Magenta = 5,

Brown = 6,

LightGray = 7,

DarkGray = 8,

LightBlue = 9,

LightGreen = 10,

LightCyan = 11,

LightRed = 12,

LightMagenta = 13,

Yellow = 14,

White = 15

};

void DrawLine();// DRAWS LINE

void WaitForPress();// WAITS FOR PRESS

void SetColor(ConsoleColor text, ConsoleColor background);// SETS TEXT AND BACKGROUND COLOR

void InputErrorMsg(); // DISPLAYS INPUT ERROR MESSAGE

1. **Шестой модуль. Реализация вспомогательных функций .**

#include "EXTRA FUNCTIONS.h"

// EXTRA FUNCTIONS

// DRAWS LINE

void DrawLine()

{

for(int i=0;i<80;i++)

cout<<'\_';

}

// WAITS FOR PRESS

void WaitForPress()

{

cout<<"\nНАЖМИТЕ ESC ДЛЯ ВОЗВРАТА...";

do{}

while (\_getch()!=27);

}

// SETS TEXT AND BACKGROUND COLOR

void SetColor(ConsoleColor text, ConsoleColor background)

{

HANDLE hConsoleOutput;

hConsoleOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsoleOutput, (WORD)((background << 4) | text));

}

// DISPLAYS INPUT ERROR MESSAGE

void InputErrorMsg()

{

if (!cin)

{

cout<<"\nОШИБКА ВВОДА!!! ПРОГРАММА НЕ ОТВЕЧАЕТ\n";

cout<<"\nНАЖМИТЕ ЛЮБУЮ КЛАВИШУ...";

\_getch();

exit(0);

}

}

**3. Результаты контрольного тестирования программы**

**1) Меню**

STUDENTS DATABASE | БАЗА ДАННЫХ О СТУДЕНТАХ УНИВЕРСИТЕТОВ ДОНЕЦКА

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1) ПЕРЕЙТИ К ВЫБОРУ УНИВЕРСИТЕТОВ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) ВВЕСТИ ИНФОРМАЦИЮ С КЛАВИАТУРЫ (С ПЕРЕЗАПИСЬЮ ТЕКУЩЕГО СПИСКА)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) ПОКАЗАТЬ СПИСОК УНИВЕРСИТЕТОВ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4) ОТКРЫТЬ ФАЙЛ (С ПЕРЕЗАПИСЬЮ ТЕКУЩЕГО СПИСКА УНИВЕРСИТЕТОВ)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5) СОХРАНИТЬ СПИСОК УНИВЕРСИТЕТОВ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6) ОПЕРАЦИИ НАД СПИСКОМ УНИВЕРСИТЕТОВ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7) УДАЛИТЬ ТЕКУЩУЮ ИНФОРМАЦИЮ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8) ЗВУКОВОЙ СИГНАЛ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ESC – ВЫХОД

**2) Ввод с клавиатуры (нажимаем 2)**

ВВЕДИТЕ КОЛИЧЕСТВО УНИВЕРСИТЕТОВ <=15:

2

ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА №1:

DonNU

ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА №2:

DonNTU

**// …**

**3) Ввод из файла и отображение на экран (нажимаем 4, а затем после возврата- 3)**

1) DonNU:

CТУДЕНТ №1: Melnik Nikolay Sergeevic Chimicheskiy Assistent 300

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №2: Prysyazhnyy Max Borysovich Fizicheskiy IT 4000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №3: Sokolov Petr Ilic Fizicheskiy Assistent 700

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №4: Yarmolnik Fedor Maximovic EPF Sekretar 6000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) DonNTU:

CТУДЕНТ №1: Povh Oleg Andreevic Fizicheskiy Laborant 3000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №2: Loginov Sergey Alexeevic Matematicheskiy Laborant 2000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАЖМИТЕ ESC ДЛЯ ВОЗВРАТА...

**4) Запись в файл и отображение (нажимаем 5)**

ФАЙЛ СОХРАНЕН!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАЖМИТЕ ESC ДЛЯ ВОЗВРАТА...

**5) Операции над списком (нажимаем 6)**

1) ПОИСК СТУДЕНТА

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) ДОБАВИТЬ УНИВЕРСИТЕТ В ПОЗИЦИЮ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ESC – ВОЗВРАТ

**5-А) Поиск элемента в списке (нажимаем 1)**

ВЫБЕРИТЕ ПОЛЕ, ПО КОТОРОМУ ВЫ ХОТИТЕ ПРОИЗВЕСТИ ПОИСК: <=2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1) ПО НАЗВАНИЮ УНИВЕРСИТЕТА

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) ПО НОМЕРУ УНИВЕРСИТЕТА

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Нажимаем 1 (будем искать по названию). Нажимаем отобразить список**

1) DonNU

2) DonNTU

ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА:

DonNTU

НАЙДЕНО В ПОЗИЦИИ: 2

2) DonNTU

НАЖМИТЕ ESC ДЛЯ ВОЗВРАТА....

**5-Б) Добавить студента в список (нажимаем 3)**

ВВЕДИТЕ ПОЗИЦИЮ В СПИСКЕ, В КОТОРУЮ СОБИРАЕТЕСЬ ДОБАВИТЬ СТУДЕНТА <=2:

2

ВВЕДИТЕ НАЗВАНИЕ УНИВЕРСИТЕТА №2:

DonNUET

**Нажимаем отобразить список**

1) DonNU:

CТУДЕНТ №1: Melnik Nikolay Sergeevic Chimicheskiy Assistent 300

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №2: Prysyazhnyy Max Borysovich Fizicheskiy IT 4000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №3: Sokolov Petr Ilic Fizicheskiy Assistent 700

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №4: Yarmolnik Fedor Maximovic EPF Sekretar 6000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) DonNUET:

ЗДЕСЬ ПОКА НЕТ НИ ОДНОЙ ЗАПИСИ...

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) DonNTU:

CТУДЕНТ №1: Povh Oleg Andreevic Fizicheskiy Laborant 3000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №2: Loginov Sergey Alexeevic Matematicheskiy Laborant 2000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАЖМИТЕ ESC ДЛЯ ВОЗВРАТА...

**6) Удалим какую либо информацию из списка (нажимаем 7)**

1) УДАЛИТЬ ИНФОРМАЦИЮ О КОНКРЕТНОМ УНИВЕРСИТЕТЕ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) УДАЛИТЬ ВСЮ ИНФОРМАЦИЮ (ОСВОБОЖДЕНИЕ ПАМЯТИ)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ESC - ВОЗВРАТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Рассмотрим на примере удаления университета из конкретной позиции (нажимаем 1)**

1) DonNU

2) DonNUET

3) DonNTU

ВВЕДИТЕ НОМЕР УНИВЕРСИТЕТА, КОТОРОГО ВЫ СОБИРАЕТЕСЬ УДАЛИТЬ ИЗ СПИСКА <=3:

**Нажимаем отобразить список**

1) DonNU:

CТУДЕНТ №1: Melnik Nikolay Sergeevic Chimicheskiy Assistent 300

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №2: Prysyazhnyy Max Borysovich Fizicheskiy IT 4000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №3: Sokolov Petr Ilic Fizicheskiy Assistent 700

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №4: Yarmolnik Fedor Maximovic EPF Sekretar 6000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2) DonNTU:

CТУДЕНТ №1: Povh Oleg Andreevic Fizicheskiy Laborant 3000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CТУДЕНТ №2: Loginov Sergey Alexeevic Matematicheskiy Laborant 2000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАЖМИТЕ ESC ДЛЯ ВОЗВРАТА...

**7) Для того чтобы изменить информацию о конкретном университете нажимаем 1. Далее ссылаюсь на лабораторную работу №3.**

**8) Чтобы подать звуковой сигнал системным динамиком нажимаем 8**

**4. Выводы**