Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

Отчёт

По учебной практике за 1 курс

Выполнил:

ст. гр. ИС/б-12-о

Долженко И.А.

Проверил:

Забаштанский А.К.

Севастополь

2018

1.Тема: Программирование операций над строками и файлами.

1.1 ПОСТАНОВКА ЗАДАЧИ

Написать программу, которая считывает текст из файла и удаляет все символы А.

1.2 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

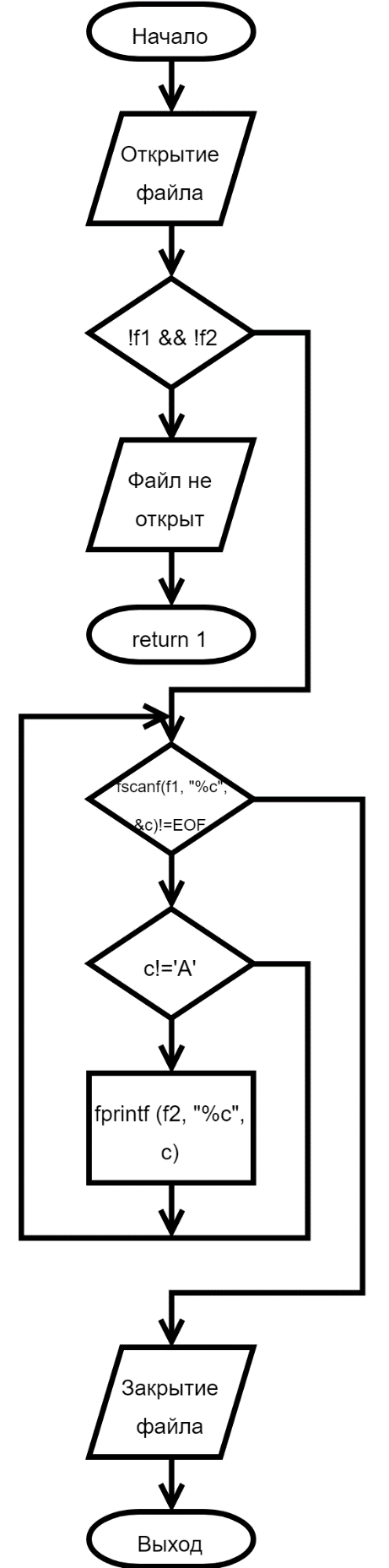


Рисунок 1 – Блок-схема программы

1.3 ТЕКСТ ПРОГРАММЫ

#include <conio.h>

#include <stdio.h>

int main()

{

char c;

FILE \*f1=fopen("in.txt", "r");

FILE \*f2=fopen("out.txt", "w");

if (!f1 && !f2)

{

puts("Can't open file\n");

return 1;

}

while (fscanf(f1, "%c", &c)!=EOF)

{

if (c!='A') fprintf (f2, "%c", c);

}

fclose(f1);

fclose(f2);

return 0;

}

1.4 ТЕСТИРОВАНИЕ

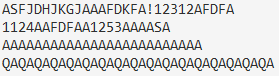


Рисунок 2 – Текст входного файла

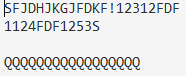


Рисунок 3 – Результат работы программы

2.Тема: Программирование операций со структурами (комбинированный тип).

2.1 ПОСТАНОВКА ЗАДАЧИ

Написать программу, которая считывает текст из файла и удаляет все символы А.

2.2 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

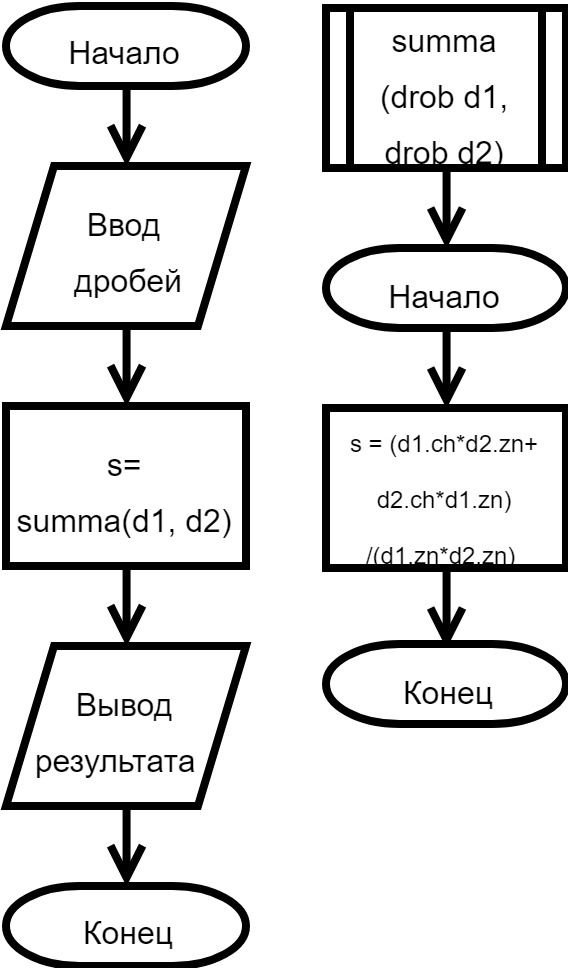


Рисунок 4 – Блок-схема программы

2.3 ТЕКСТ ПРОГРАММЫ

#include <iostream>

#include <conio.h>

using namespace std;

struct drob {

int ch, zn;

};

float summa (drob d1, drob d2) {

float s = (d1.ch\*d2.zn+d2.ch\*d1.zn)/(d1.zn\*d2.zn);

return s;

}

int main() {

float s=0;

struct drob d1; struct drob d2;

cout<<"Vvedite ch. 1 drobi:"; cin>>d1.ch;

cout<<"Vvedite zn. 1 drobi:"; cin>>d1.zn;

cout<<"Vvedite ch. 2 drobi:"; cin>>d2.ch;

cout<<"Vvedite zn. 2 drobi:"; cin>>d2.zn;

s=summa(d1, d2);

cout<<"Result:"<<s;

return 0;

}

2.4 ТЕСТИРОВАНИЕ

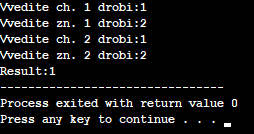


Рисунок 5 – Тест маленьких дробей

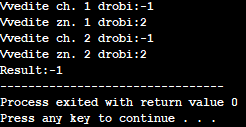


Рисунок 6 – Тест отрицательных дробей

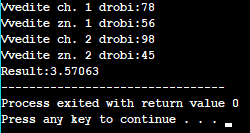


Рисунок 7 – Тест больших дробей

3.Тема: Программирование линейных списков на языке С.

3.1 ПОСТАНОВКА ЗАДАЧИ

Представить таблицу в виде линейного списка L, элементами которого являются строки таблицы.

Таблица 3.1 – Список студентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Фамилия  студента | Оценки | | |
| Физика | История | Математика |
|  |  |  |  |  |

Написать программу, содержащую следующие функции:

* создание списка (ввод данных с клавиатуры)
* просмотр списка (вывод данных в табличном виде)
* добавление нового элемента в список
* сохранения данных из списка в файл
* загрузка данных из файла и создание списка на их основе
* удаление всех элементов списка

Оформить программу в виде меню. Каждый пункт меню отвечает одной из функций. Дополнить программу функцией в соответствии с вариантом задания. Предусмотреть все проверки.

Дополнительная функция: функция подсчета средней успеваемости группы по каждой дисциплине.

3.2 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

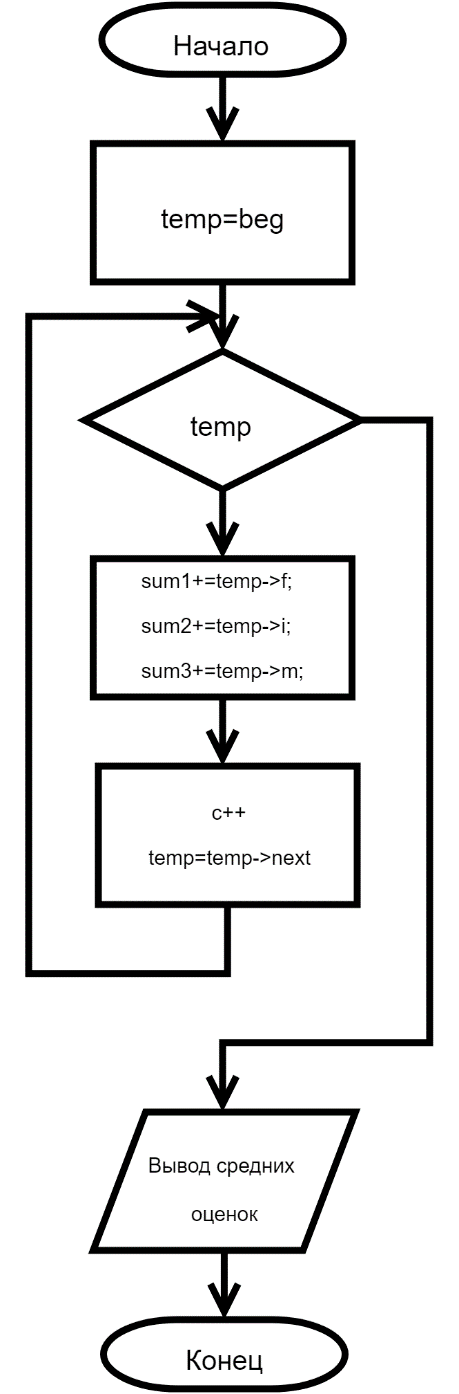


Рисунок 8 – Блок-схема функций

3.3 ТЕКСТ ПРОГРАММЫ

#include <fstream>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <time.h>

#include <iomanip>

#include <conio.h>

#include <iostream>

using namespace std;

const int len=20;

struct stud

{

char fio[len];

float f, i, m;

stud\* next;

};

stud\* dob(stud\* end, const stud& z);

stud\* dob\_first(const stud& z);

stud\* udal(stud\* beg);

stud vvod\_studa();

int menu();

void print(const stud& z);

void prosmotr(stud\* beg);

int read\_file(char\* filename, stud\*\* beg, stud\*\* end);

int write\_file(char\* filename, stud\* temp);

void ocenki(stud \*beg);

int main()

{

setlocale(0,"RUS");

stud \*beg=0, \*end=0;

char \*filename=(char \*)"123.txt";

read\_file(filename,&beg, &end);

while (1)

{

switch (menu())

{

case 1: if (beg) end=dob(end,vvod\_studa());

else

{

beg=dob\_first(vvod\_studa());

end=beg;

}

break;

case 2: beg=udal(beg);

cout<<"Нажмите любую клавишу"<<endl;

cin.get();

break;

case 3: prosmotr(beg);

break;

case 4: ocenki(beg);

break;

case 5: write\_file(filename,beg);

break;

case 6: return 0;

default:

cout<<"Вам следует ввести номер от 1 до 7"<< endl;

cin.get();

break;

}

}

}

stud\* dob(stud \*end,const stud& z)

{

stud \*newE=new stud;

\*newE=z;

newE->next=0;

end->next=newE;

end=newE;

return end;

}

stud\* dob\_first(const stud& z)

{

stud \*beg=new stud;

\*beg=z;

beg->next=0;

return beg;

}

stud\* udal(stud \*beg)

{

stud \*p=beg;

while (p->next)

{

stud \*temp;

temp = p;

p = p->next;

delete temp;

}

p=p->next;

delete []p;

beg=NULL;

return beg;

}

stud vvod\_studa()

{

char buf[10];

stud z;

cout<<"Введите Ф.И.О.:"<<endl;

cin.getline(z.fio,len);

cout<<"Введите оценку по Физике:"<<endl;

cin>>z.f;

cout<<"Введите оценку по Истории:"<<endl;

cin>>z.i;

cout<<"Введите оценку по Математике:"<<endl;

cin>>z.m;

return z;

}

int menu()

{

char buf[10];

int item;

do

{

system("CLS");

cout<<"1- Добавление элемента в список"<<endl;

cout<<"2- Удаление всех элементов"<<endl;

cout<<"3- Просмотр списка"<<endl;

cout<<"4- Успеваемость по предметам"<<endl;

cout<<"5- Запись данных в файл"<<endl;

cout<<"6- Выход"<<endl;

cout<<"============================"<<endl;

cout<<"Введите номер пункта меню"<<endl;

cin>>buf;

cin.get();

item=atoi(buf);

if (!item)

{

cout<<"Вам следует вводить число от 1 до 6"<<endl;

cin.get();

}

}

while (!item);

return item;

}

void print(const stud& z)

{

cout<<"Ф.И.О.: " <<z.fio<<endl;

cout<<"Оценка по Физике: "<<z.f<<endl;

cout<<"Оценка по Истории: "<<z.i<<endl;

cout<<"Оценка по Математике: "<<z.m<<endl;

}

void prosmotr(stud \*beg)

{

if (!beg)

{

cout<<"Очередь пустая"<<endl;

return;

}

stud \*temp=beg;

cout<<"=============================="<<endl;

while (temp)

{

print(\*temp);

cout<<"=============================="<<endl;

cout<<"Нажмите любую клавишу"<<endl;

cin.get();

temp=temp->next;

}

}

int read\_file(char\* filename,stud\*\* beg, stud\*\* end)

{

ifstream fin(filename,ios::in);

if (!fin) { cout<<"Нет файла"<<filename<<endl; return 1;}

stud z;

\*beg = 0;

while (fin.getline(z.fio,len))

{

fin>>z.f;

fin>>z.i;

fin>>z.m;

fin.get();

if (\*beg) \*end=dob(\*end,z);

else

{ \*beg=dob\_first(z); \*end=\*beg; }

}

return 0;

}

int write\_file(char\* filename, stud\* temp)

{

ofstream fout(filename);

if (!fout) {cout<<"Не могу открыть файл"<<endl; return 1;}

while (temp)

{

fout<<temp->fio<<endl;

fout<<temp->f<<endl;

fout<<temp->i<<endl;

fout<<temp->m<<endl;

temp=temp->next;

}

cout<<"Данные сохранены в файле: "<<filename<<endl;

cout<<"=============================="<<endl;

cout<<"Нажмите любую клавишу"<<endl;

cin.get();

return 0;

}

void ocenki(stud \*beg)

{

float sum1, sum2, sum3, c=0;

stud \*temp;

temp=beg;

while (temp)

{

sum1+=temp->f;

sum2+=temp->i;

sum3+=temp->m;

c++;

temp=temp->next;

}

cout<<"Средняя оценка по Физике:"<<sum1/c<<endl;

cout<<"Средняя оценка по Истории:"<<sum2/c<<endl;

cout<<"Средняя оценка по Математике:"<<sum3/c<<endl;

cin.get();

}

4.4 ТЕСТИРОВАНИЕ

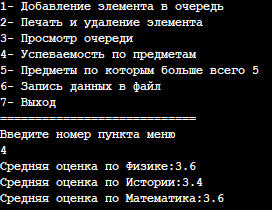


Рисунок 9 – Средние оценки по дисциплинам

4.Тема: Программирование задач “Длинной арифметики”.

4.1 ПОСТАНОВКА ЗАДАЧИ

Даны два числа в 200 знаков каждое. Необходимо вычислить сумму этих двух чисел.

4.2 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

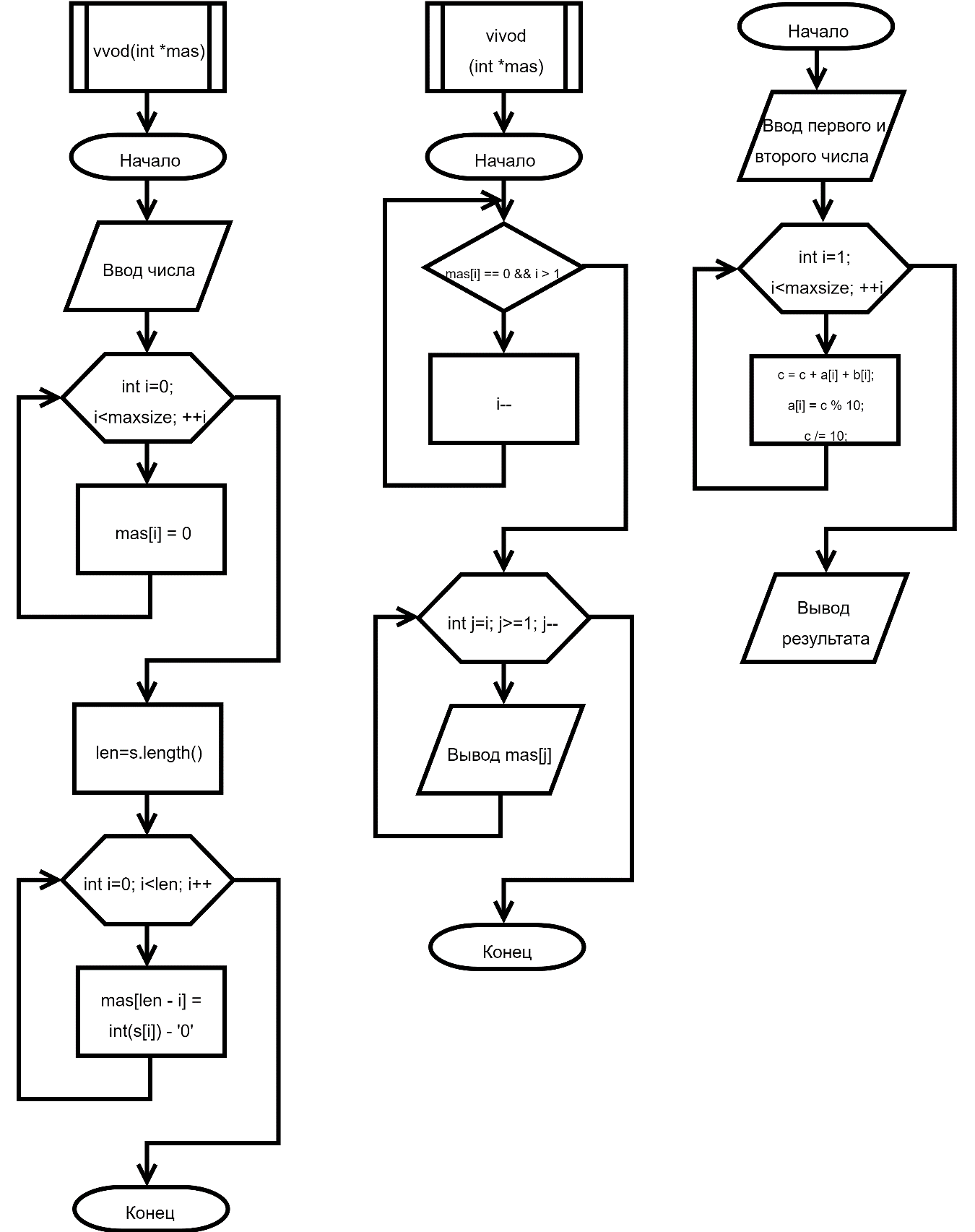


Рисунок 10 – Блок-схема функций

4.3 ТЕКСТ ПРОГРАММЫ

#include <iostream>

using namespace std;

const int maxsize = 100;

void vvod(int \*mas)

{

string s;

cin >> s;

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

mas[i] = 0;

int len;

len=s.length();

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

mas[len - i] = int(s[i]) - '0';

}

void vivod(int \*mas)

{

int i = maxsize;

while(mas[i] == 0 && i > 1) i--;

for(int j=i; j>=1; j--) cout << mas[j];

}

int main()

{

setlocale(0, "rus");

int a[maxsize], b[maxsize];

cout << "Введите первое число:" << endl;

vvod(a);

cout << "Введите второе число:" << endl;

vvod(b);

int c = 0;

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

{

c = c + a[i] + b[i];

a[i] = c % 10;

c /= 10;

}

cout << "Результат:" << endl;

vivod(a);

}

4.4 ТЕСТИРОВАНИЕ

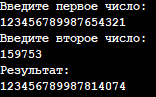


Рисунок 11 – Тест длинных чисел

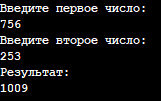


Рисунок 12 – Тест коротких чисел

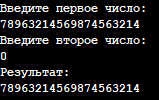


Рисунок 13 – Тест с 0

5.Тема: Программирование нелинейных динамических структур.

5.1 ПОСТАНОВКА ЗАДАЧИ

Представить таблицу в виде бинарного дерева, элементами которого являются строки таблицы.

Таблица 5.1 – Список студентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Фамилия  студента | Оценки | | |
| Физика | История | Математика |
|  |  |  |  |  |

Написать программу, содержащую следующие функции:

* создание дерева (ввод данных с клавиатуры)
* просмотр дерева (вывод данных в табличном виде)
* отображение дерева (вывод структуры дерева)
* сохранение данных в файл
* загрузка данных из файла
* удаление всех элементов дерева

Оформить программу в виде меню. Каждый пункт меню отвечает одной из функций. Дополнить программу функцией в соответствии с вариантом задания. Предусмотреть все проверки.

Дополнительная функция: функция подсчета количества отличников.

5.2 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

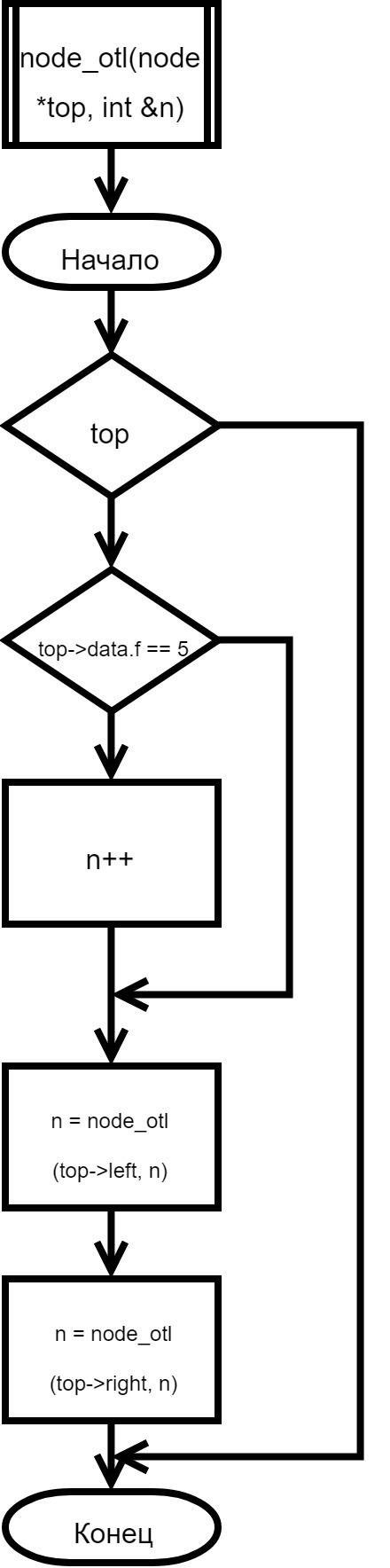


Рисунок 14 – Блок-схема функции

5.3 ТЕКСТ ПРОГРАММЫ

#include <fstream>

#include <cstdlib>

#include <cstring>

#include <iomanip>

#include <conio.h>

#include <iostream>

#include <windows.h>

using namespace std;

const int d\_f = 20;

struct stud

{

char fio[d\_f];

int f, i, m;

};

struct node

{

stud data;

node\* left;

node\* right;

};

node\* addtree(node \*top, const stud& newnode);

int menu();

int node\_otl(node \*top, int &n);

void otobr(node \*top, int otstup);

void prosmotr(node \*top);

int write\_file(ofstream &f, node\* top);

int read\_file(char\* filename, node\* &top);

stud vvod();

void FreeTree(node \* top)

{

if (top)

{

if (top->left) FreeTree(top->left);

if (top->right) FreeTree(top->right);

delete top;

}

}

int main()

{

setlocale(0, "Russian");

node \*top = 0;

char \*filename = (char\*) ("stud.txt");

ofstream fout;

read\_file(filename, top);

while (1)

{

switch (menu())

{

case 1:

{

top = addtree(top, vvod());

break;

}

case 2:

{

otobr(top, 1);

cout << "Нажмите любую клавишу " << endl;

cin.get();

break;

}

case 3:

{

FreeTree(top);

break;

}

case 4:

{

cout << " Имя | Физика | История | Математика |" << endl;

prosmotr(top);

cout << "Нажмите любую клавишу " << endl;

cin.get();

break;

}

case 5:

{

int n = 0;

cout << node\_otl(top, n) << " отличников" << endl;

cout << "Нажмите любую клавишу " << endl;

cin.get();

break;

}

case 6:

{

fout.open(filename);

if (!fout) {

cout << "Ошибка открытия файла" << endl; return 1;

}

write\_file(fout, top);

cout << "Данные сохранены в файле: " << filename << endl;

cout << "==============================" << endl;

fout.close();

cout << "Нажмите любую клавишу " << endl;

cin.get();

break;

}

case 7: return 0;

default:

{

cout << "Вам следует ввести число от 1 до 7" << endl;

cin.get();

break;

}

}

}

}

node\* addtree(node \*top, const stud& newnode)

{

if (!top)

{

top = new node;

if (!top) {

cout << "Не хватает памяти" << endl;

return NULL;

}

top->data = newnode;

top->left = NULL;

top->right = NULL;

}

else

if (strcmp(top->data.fio, newnode.fio)>0)

top->left = addtree(top->left, newnode);

else

top->right = addtree(top->right, newnode);

return top;

}

void otobr(node \*top, int otstup)

{

if (top)

{

otstup += 3;

otobr(top->right, otstup);

cout << setw(otstup)<< '\*' << top->data.fio << endl;

otobr(top->left, otstup);

}

}

void prosmotr(node \*top)

{

if (top)

{

cout << setw(12) << left << top->data.fio;

cout << setw(10) << left << top->data.f;

cout << setw(13) << left << top->data.i;

cout << setw(18) << left << top->data.m << endl;

prosmotr(top->left);

prosmotr(top->right);

}

}

stud vvod()

{

stud p;

cout << "Введите имя:" << endl;

cin.getline(p.fio, d\_f);

cout << "Введите оценку по Физике:" << endl;

cin >> p.f;

cout << "Введите оценку по Истории:" << endl;

cin >> p.i;

cout << "Введите оценку по Математике:" << endl;

cin >> p.m;

return p;

}

int menu()

{

char key = 0;

int point = 0;

do

{

system("cls");

if (key==72)

{

if (--point<0) point=7;

}

else if (key==80)

{

if (++point>7) point=0;

}

cout << ((point==1)?'>':' ')<< "1- Добавить элемент в дерево" << endl;

cout << ((point==2)?'>':' ')<< "2- Отобразить структуру дерева" << endl;

cout << ((point==3)?'>':' ')<< "3- Удаление всех элементов дерева" << endl;

cout << ((point==4)?'>':' ')<< "4- Просмотр дерева" << endl;

cout << ((point==5)?'>':' ')<< "5- Подсчет количества отличников" << endl;

cout << ((point==6)?'>':' ')<< "6- Запись данных в файл" << endl;

cout << ((point==7)?'>':' ')<< "7- Выход" << endl;

key=getch();

} while (key!=13 && key!=27);

system ("cls");

return key==13?point:6;

}

int node\_otl(node \*top, int &n)

{

if (top)

{

if (top->data.f == 5 && top->data.i == 5 && top->data.m == 5) n++;

n = node\_otl(top->left, n);

n = node\_otl(top->right, n);

}

return n;

}

int read\_file(char\* filename, node\* &top)

{

ifstream fin(filename, ios::in);

if (!fin) { cout << "Не найден файл" << filename << endl; return 1; }

stud p;

top = 0;

while (fin.getline(p.fio, d\_f))

{

fin>>p.f;

fin>>p.i;

fin>>p.m;

top = addtree(top, p);

}

return 0;

}

int write\_file(ofstream &f, node\* top)

{

if (top)

{

f << top->data.fio << endl;

f << top->data.f << endl;

f << top->data.i << endl;

f << top->data.m << endl;

write\_file(f, top->left);

write\_file(f, top->right);

}

return 0;

}

5.4 ТЕСТИРОВАНИЕ

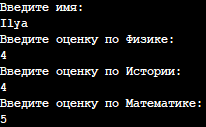


Рисунок 15 – Ввод данных

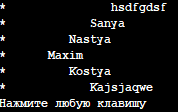


Рисунок 16 – Отображение дерева

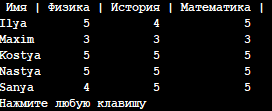


Рисунок 17 – Просмотр дерева



Рисунок 18 – Поиск отличников

6.Тема: Реализация динамического интерфейса.

6.1 ПОСТАНОВКА ЗАДАЧИ

Необходимо написать программу таким образом, чтобы интерфейс программы стал интерактивным, т.е. необходимый пункт меню выбирался с помощью управляющих клавиш (стрелочки) и запускался на выполнение нажатием клавиши Enter. Текущий пункт меню можно выделять подсветкой цветом или с помощью некоего “курсора” – вспомогательного символа.

6.2 СТРУКТУРНАЯ СХЕМА АЛГОРИТМА

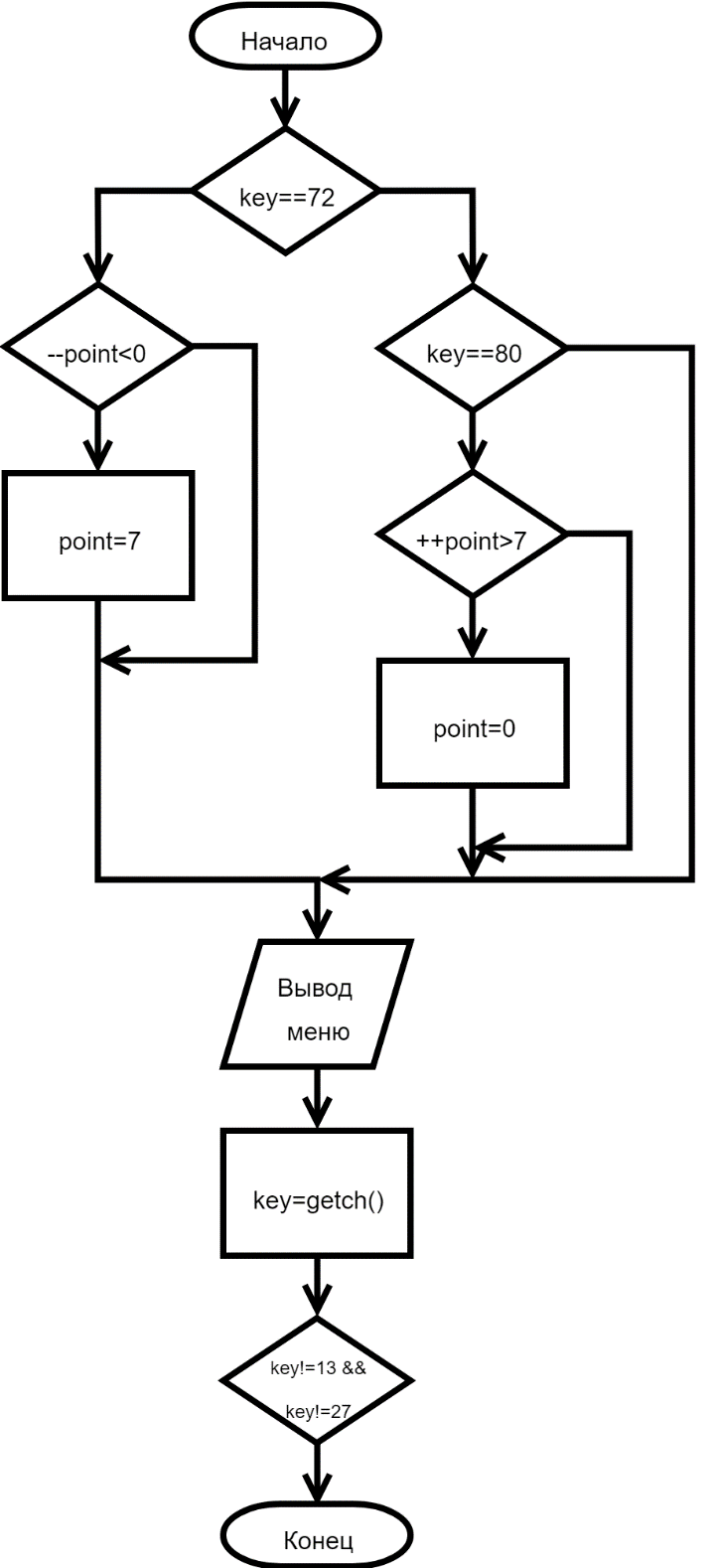


Рисунок 19 – Блок-схема программы

6.3 ТЕКСТ ПРОГРАММЫ

int menu()

{

char key = 0;

int point = 0;

do

{

system("cls");

if (key==72)

{

if (--point<0) point=7;

}

else if (key==80)

{

if (++point>7) point=0;

}

cout << ((point==1)?'>':' ')<< "1- Добавить элемент в дерево" << endl;

cout << ((point==2)?'>':' ')<< "2- Отобразить структуру дерева" << endl;

cout << ((point==3)?'>':' ')<< "3- Удаление всех элементов дерева" << endl;

cout << ((point==4)?'>':' ')<< "4- Просмотр дерева" << endl;

cout << ((point==5)?'>':' ')<< "5- Подсчет количества отличников" << endl;

cout << ((point==6)?'>':' ')<< "6- Запись данных в файл" << endl;

cout << ((point==7)?'>':' ')<< "7- Выход" << endl;

key=getch();

} while (key!=13 && key!=27);

system ("cls");

return key==13?point:7;

}

6.4 ТЕСТИРОВАНИЕ

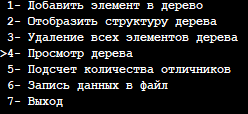


Рисунок 20 – Интерактивное меню

ВЫВОД

В ходе выполнения практической работы были повторены операции над строками, файлами и структурами. Были получены навыки работы с длинной арифметикой. Были написаны программы реализации таблицы студентов в виде списка и дерева, содержащие следующие функции: создание, просмотр, сохранения данных в файл, удаление всех элементов. Также был реализован динамический интерфейс программы, в котором пункты меню выбираются стрелочками.