Национальный исследовательский университет «МИЭТ»

Отчет о выполненной лабораторной работе №7

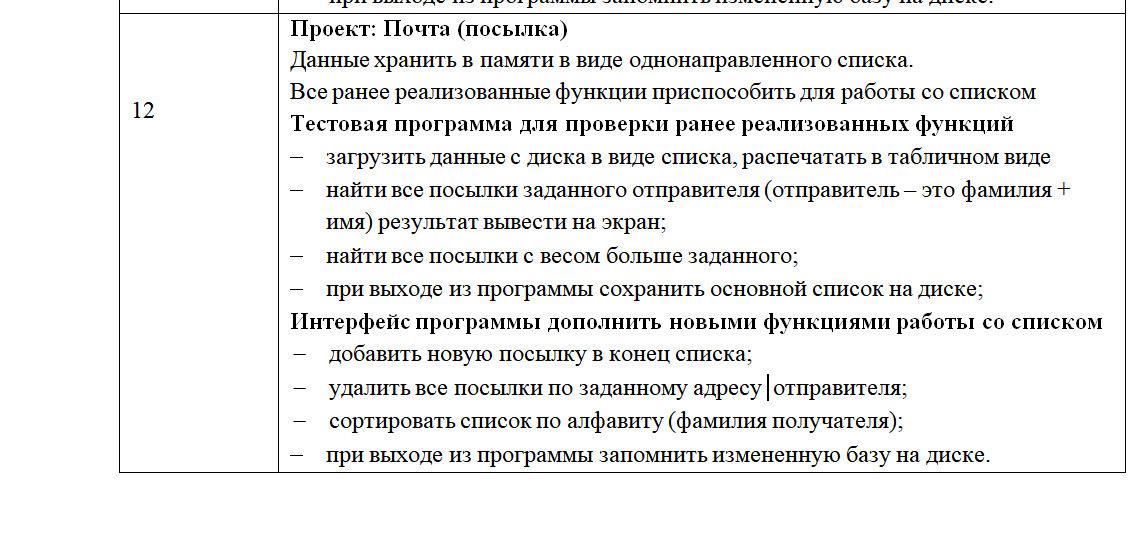
По предмету :Практикум по программированию на языке С

Выполнила Марина Алина

Группа ПИН-14

2020г.

Данный вариант придуман по аналогии с другими вариантами в 7 лабораторной работе



#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_NONSTDC\_NO\_DEPRECATE

#include <stdlib.h>

#include <stdio.h>

#include <windows.h>

#include <iostream>

#include <fstream>

#include <iomanip>

#define l\_buff 60

#define s\_buff 80

#define N 40

using namespace std;

struct parcel

{

char\* surname;// фамилия отправителя

char\* name;// имя отправителя

char\* adres;// адрес отправителя

char\* surname1;//фамилия получателя

char\* name1;// имя получателя

double weight;// вес

} bebe;

typedef struct List

{

List\* Next;

parcel date;

}List;

//Добавляет новую посылку в базу

void add\_new\_parcel(List\* head)

{

char\* str\_buff = new char[s\_buff];

if ((head->Next == 0) && (head->date.name1 == NULL))

{

List\* S = new List;

cout << "Введите фамилию отправителя: ";

cin >> str\_buff;

S->date.surname = new char[strlen(str\_buff) + 1];

strcpy\_s(S->date.surname, strlen(str\_buff) + 1, str\_buff);

cout << "Введите имя отправителя: ";

cin >> str\_buff;

S->date.name = new char[strlen(str\_buff) + 1];

strcpy\_s(S->date.name, strlen(str\_buff) + 1, str\_buff);

cout << "Введите адрес отправителя: ";

cin >> str\_buff;

S->date.adres = new char[strlen(str\_buff) + 1];

strcpy\_s(S->date.adres, strlen(str\_buff) + 1, str\_buff);

cout << "Введите фамилию получателя: ";

cin >> str\_buff;

S->date.surname1 = new char[strlen(str\_buff) + 1];

strcpy\_s(S->date.surname1, strlen(str\_buff) + 1, str\_buff);

cout << "Введите имя получателя: ";

cin >> str\_buff;

S->date.name1 = new char[strlen(str\_buff) + 1];

strcpy\_s(S->date.name1, strlen(str\_buff) + 1, str\_buff);

cout << "Введите вес посылки: ";

cin >> S->date.weight;

cout << "\n\n";

head->date = S->date;

head->Next = NULL;

}

else

{

List\* New = new List;

List\* Cur = head;

New->Next = NULL;

cout << "Введите фамилию отправителя: ";

cin >> str\_buff;

New->date.surname = new char[strlen(str\_buff) + 1];

strcpy\_s(New->date.surname, strlen(str\_buff) + 1, str\_buff);

cout << "Введите имя отправителя: ";

cin >> str\_buff;

New->date.name = new char[strlen(str\_buff) + 1];

strcpy\_s(New->date.name, strlen(str\_buff) + 1, str\_buff);

cout << "Введите адрес отправителя: ";

cin >> str\_buff;

New->date.adres = new char[strlen(str\_buff) + 1];

strcpy\_s(New->date.adres, strlen(str\_buff) + 1, str\_buff);

cout << "Введите фамилию получателя: ";

cin >> str\_buff;

New->date.surname1 = new char[strlen(str\_buff) + 1];

strcpy\_s(New->date.surname1, strlen(str\_buff) + 1, str\_buff);

cout << "Введите имя получателя: ";

cin >> str\_buff;

New->date.name1 = new char[strlen(str\_buff) + 1];

strcpy\_s(New->date.name1, strlen(str\_buff) + 1, str\_buff);

cout << "Введите вес посылки: ";

cin >> New->date.weight;

cout << "\n\n";

while (Cur->Next != 0)

Cur = Cur->Next;

Cur->Next = New;

}

}

//красивый вывод

int vivod(double a)

{

char b[15];

int n = 10;

itoa(a, b, n);

return strlen(b);

}

//Выводит список всех посылок из базы

void print\_all\_parcel (List\* head)

{

cout << "Информация о посылках :" << endl;

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

cout << " Фамилия и имя отправителя " << "Адрес отправителя " << " Фамилия и имя получателя " << " Вес посылки " << endl;

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

if (head->date.name1 == 0)

cout << " Пусто " << endl;

else

{

List\* Cur;//рабочий указатель

for (Cur = head; Cur->Next != 0; Cur = Cur->Next)

{

int tr1 = (30 - strlen(Cur->date.surname) - strlen(Cur->date.name) - 1) / 2;

int tr2 = (19-strlen(Cur->date.adres) + 20) / 2;

int tr3 = (29 - strlen(Cur->date.surname1) - strlen(Cur->date.name1) - 1) / 2;

int tr4 = (-vivod(Cur->date.weight) + 20) / 2;

cout << ' ' << setfill(' ') << setw(tr1) << ' ' << Cur->date.surname << ' ' << Cur->date.name << ' ' << setfill(' ') << setw(tr1) << ' '

<< ' ' << setfill(' ') << setw(tr2) << ' ' << Cur->date.adres << ' ' << setfill(' ') << setw(tr2) << ' '

<< ' ' << setfill(' ') << setw(tr3) << ' ' << Cur->date.surname1 << ' ' << Cur->date.name1 << ' ' << setfill(' ') << setw(tr3) << ' '

<< ' ' << setfill(' ') << setw(tr4) << ' ' << Cur->date.weight << ' ' << setfill(' ') << setw(tr4) << ' ' << endl

<< "-----------------------------------------------------------------------------------------------------------------------" << endl;

}

//Для последнего письма в списке

int tr1 = (30 - strlen(Cur->date.surname) - strlen(Cur->date.name) - 1) / 2;

int tr2 = (19-strlen(Cur->date.adres) + 20) / 2;

int tr3 = (29 - strlen(Cur->date.surname1) - strlen(Cur->date.name1) - 1) / 2;

int tr4 = (-vivod(Cur->date.weight) + 20) / 2;

cout << ' ' << setfill(' ') << setw(tr1) << ' ' << Cur->date.surname << ' ' << Cur->date.name << ' ' << setfill(' ') << setw(tr1) << ' '

<< ' ' << setfill(' ') << setw(tr2) << ' ' << Cur->date.adres << ' ' << setfill(' ') << setw(tr2) << ' '

<< ' ' << setfill(' ') << setw(tr3) << ' ' << Cur->date.surname1 << ' ' << Cur->date.name1 << ' ' << setfill(' ') << setw(tr3) << ' '

<< ' ' << setfill(' ') << setw(tr4) << ' ' << Cur->date.weight << ' ' << setfill(' ') << setw(tr4) << ' ' << endl

<< "-----------------------------------------------------------------------------------------------------------------------" << endl;

}

cout << endl;

}

//Выводит список посылок определённого отправителя

void print\_parcel\_ot(List\* head)

{

char\* str\_buff1 = new char[s\_buff];

char\* str\_buff2 = new char[s\_buff];

int ln = 0;

cout << "Введите фамилию: ";

cin >> str\_buff1;

cout << "Введите имя: ";

cin >> str\_buff2;

cout << "\nНайденны посылки, отправитель которых " << str\_buff1 << ' ' << str\_buff2 << endl;

cout << "Информация о посылках :" << endl;

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

cout << " Адрес отправителя " << " Фамилия и имя получателя " << " Вес посылки " << endl;

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

if (head->date.surname1 == 0)

cout << " База писем пуста" << endl;

else

{

List\* Cur;

for (Cur = head; Cur->Next != 0; Cur = Cur->Next)

{

if (strcmp(str\_buff1, Cur->date.surname) == 0)

{

ln++;

if (strcmp(str\_buff2, Cur->date.name) == 0)

{

int tr2 = (20-strlen(Cur->date.adres) + 5) / 2;

int tr3 = (19 - strlen(Cur->date.surname1) - strlen(Cur->date.name1) - 1) / 2;

int tr4 = (-vivod(Cur->date.weight) + 34) / 2;

cout << ' ' << setfill(' ') << setw(tr2) << ' ' << Cur->date.adres << ' ' << setfill(' ') << setw(tr2) << ' '

<< ' ' << setfill(' ') << setw(tr3) << ' ' << Cur->date.surname1 << ' ' << Cur->date.name1 << ' ' << setfill(' ') << setw(tr3) << ' '

<< ' ' << setfill(' ') << setw(tr4) << ' ' << Cur->date.weight << ' ' << setfill(' ') << setw(tr4) << ' ' << endl

<< "-------------------------------------------------------------------------------------" << endl;

}

}

}

//Для последнего элемента в списке

if (strcmp(str\_buff1, Cur->date.surname) == 0)

{

ln++;

if (strcmp(str\_buff2, Cur->date.name) == 0)

{

int tr2 = (20-strlen(Cur->date.adres) + 5) / 2;

int tr3 = (19 - strlen(Cur->date.surname1) - strlen(Cur->date.name1) - 1) / 2;

int tr4 = (-vivod(Cur->date.weight) + 34) / 2;

cout << ' ' << setfill(' ') << setw(tr2) << ' ' << Cur->date.adres << ' ' << setfill(' ') << setw(tr2) << ' '

<< ' ' << setfill(' ') << setw(tr3) << ' ' << Cur->date.surname1 << ' ' << Cur->date.name1 << ' ' << setfill(' ') << setw(tr3) << ' '

<< ' ' << setfill(' ') << setw(tr4) << ' ' << Cur->date.weight << ' ' << setfill(' ') << setw(tr4) << ' ' << endl

<< "--------------------------------------------------------------------------------------" << endl;

}

}

if (ln == 0)

cout << " У отправителя нет посылок" << endl;

}

cout << endl;

}

//Добавляет новую посылку в конец списка(без интерфейса)

void add\_new\_parcel\_konec(List\* head, List\* Date)

{

if ((head->Next == 0) && (head->date.name1 == NULL))

{

head->date = Date->date;

}

else

{

List\* Cur = head;

List\* New = new List;

New->date = Date->date;

New->Next = NULL;

while (Cur->Next != NULL)

Cur = Cur->Next;

Cur->Next = New;

}

}

//Создаёт список посылок с весом больше заданного

List\* filtr\_weight(List\* head)

{

double weight;

List\* New = new List;

New->Next = NULL;

New->date.name1 = NULL;

List\* Cur;

List\* CurN = New;

cout << "Введите вес , с которым нужно сравнить: ";

cin >> weight;

for (Cur = head; Cur->Next != 0; Cur = Cur->Next)

{

if (Cur->date.weight >= weight)

{

add\_new\_parcel\_konec(New, Cur);

}

}

if (Cur->date.weight >= weight)

{

add\_new\_parcel\_konec(New, Cur);

cout << "Все хорошо!" << endl;

}

return New;

}

//Удаляет все письма по заданному адресу отправителя

void delete\_parcel(List\*\* head)

{

char adres1[N];

cout << "\nВведите адрес: ";

cin >> adres1;

List\* Cur, \* Work;

int i = 0;

while (i == 0)

{

i = 1;

if ((\*head)->date.adres == adres1)

{

Work = (\*head)->Next;

if (!Work)

{

(\*head)->date.name1 = NULL;

(\*head)->date.adres = NULL;

i = 0;

}

else

{

delete[] \* head;

(\*head) = Work;

i = 0;

}

}

else

{

Cur = \*head;

while (Cur->Next != NULL)

{

if (Cur->Next->date.adres != adres1)

Cur = Cur->Next;

else

{

i = 0;

Work = Cur->Next->Next;

delete[](Cur->Next);

Cur->Next = Work;

}

}

}

}

cout <<"Сделано. Вам стоит выйти из программы с сохранением изменений и потом запустить заново, если хотите проверить"<< endl;

}

//Сортирует посылки по алфавиту (фамилии получателя)

void sort\_alfavit(List\*\* head)

{

int flag = 0;

List\* Cur;

while (flag == 0)

{

flag = 1;

Cur = \*head;

while (Cur->Next != NULL)

{

if (strcmp(Cur->date.surname1, Cur->Next->date.surname1) > 0)

{

flag = 0;

List\* B = new List;

B->date = Cur->Next->date;

(Cur->Next)->date = Cur->date;

Cur->date = B->date;

}

Cur = Cur->Next;

}

}

cout << "\nПосылки отсортированы\n" << endl;

}

//Загружает базу (или создаёт при её отсутствии) посылок из текстового файла на диске список, путь указывает пользователь

void download\_parcel(List\*\* head, char\* path)

{

char\* buff = new char[s\_buff];

cout << "\n(Пример ввода C:/visual\_studio/text.txt)\nВведите путь к файлу: ";

cin >> buff;

strcpy(path, buff);

ifstream inf(path);

if (!inf)

{

inf.close();

ofstream outf(path);

cout << "\nФайл создан\n" << endl;

outf.close();

}

else

{

cout << "\nФайл найден\n" << endl;

List\* temp = new List;

temp->Next = 0;

while (!inf.eof())

{

double w;

int iS, iR;

char\* sp = new char[5];

inf >> w >> iS;

inf.getline(sp, 3, ' ');

char\* buff = new char[s\_buff];

inf.getline(buff, s\_buff + 10, '|');

temp->date.surname = new char[strlen(buff) + 1];

strcpy\_s(temp->date.surname, strlen(buff) + 1, buff);

inf.getline(buff, s\_buff + 10, '|');

temp->date.name = new char[strlen(buff) + 1];

strcpy\_s(temp->date.name, strlen(buff) + 1, buff);

inf >> iR;

inf.getline(sp, 3, ' ');

inf.getline(buff, s\_buff + 10, '|');

temp->date.adres = new char[strlen(buff) + 1];

strcpy\_s(temp->date.adres, strlen(buff) + 1, buff);

inf.getline(buff, s\_buff + 10, '|');

temp->date.surname1 = new char[strlen(buff) + 1];

strcpy\_s(temp->date.surname1, strlen(buff) + 1, buff);

inf.get(buff, s\_buff + 10);

temp->date.name1 = new char[strlen(buff) + 1];

strcpy\_s(temp->date.name1, strlen(buff) + 1, buff);

inf.getline(buff, s\_buff + 10, '|');

temp->date.weight = w;

add\_new\_parcel\_konec(\*head, temp);

}

}

}

//Сохраняет базу посылок

void save\_parcel(List\* head, char\* path)

{

List\* Cur = head;

char\* str\_buff = new char[s\_buff];

char\* path2 = new char[s\_buff];

if (\*path != NULL)

{

cout << "\nСохранить в " << path << " ? Напишите да или нет" << endl;

cin >> str\_buff;

char y[] = "да\0";

char n[] = "нет\0";

if ((strcmp(str\_buff, y) != 0) && (strcmp(str\_buff, n) != 0))

save\_parcel(head, path);

if (strcmp(str\_buff, y) == 0)

{

ofstream outf(path);

while (Cur->Next != 0)

{

outf << Cur->date.surname << ' ' << Cur->date.name << '|' << Cur->date.adres << '|'

<< Cur->date.surname1 << ' ' << Cur->date.name1 << '|' << Cur->date.weight << endl;

Cur = Cur->Next;

}

outf << Cur->date.surname << ' ' << Cur->date.name << '|' << Cur->date.adres << '|'

<< Cur->date.surname1 << ' ' << Cur->date.name1 << '|' << Cur->date.weight << endl;

}

if (strcmp(str\_buff, n) == 0)

{

cout << "\n(Пример ввода C:/visual\_studio/text.txt)\nВведите путь к файлу: ";

cin >> path2;

ofstream outf(path2);

while (Cur->Next != 0)

{

outf << Cur->date.surname << ' ' << Cur->date.name << '|' << Cur->date.adres << '|'

<< Cur->date.surname1 << ' ' << Cur->date.name1 << '|' << Cur->date.weight << endl;

Cur = Cur->Next;

}

if (Cur->date.name1)

{

outf << Cur->date.surname << ' ' << Cur->date.name << '|' << Cur->date.adres << '|'

<< Cur->date.surname1 << ' ' << Cur->date.name1 << '|' << Cur->date.weight << endl;

}

}

}

else

{

cout << "\n(Пример ввода C:/visual\_studio/text.txt)\nВведите путь к файлу: "; cin >> path2;

ofstream outf(path2);

while (Cur->Next != 0)

{

outf << Cur->date.surname << ' ' << Cur->date.name << '|'<< Cur->date.adres << '|'

<< Cur->date.surname1 << ' ' << Cur->date.name1 <<'|'<< Cur->date.weight << endl;

Cur = Cur->Next;

}

if (Cur->date.name1)

{

outf << Cur->date.surname << ' ' << Cur->date.name << '|' << Cur->date.adres << '|'

<< Cur->date.surname1 << ' ' << Cur->date.name1 << '|' << Cur->date.weight << endl;

}

}

}

//Меню

void menu(List\* head)

{

bool v = true;

cout << "\t\t\tПроект почта(посылка)" << endl;

char\* pth\_file = new char[s\_buff];

\*pth\_file = NULL;

List\* filtra = new List;

List\*\* H;

H = &head;

while (v)

{

int p;

cout << "Добавить новую посылку .................................1" << endl;

cout << "Распечатать информацию о посылках в табличном виде .....2" << endl;

cout << "Показать все посылки заданного отправителя .............3" << endl;

cout << "Показать все письма с весом больше заданного ...........4" << endl;

cout << "Удалить все посыки по заданному адресу отправителя .....5" << endl;

cout << "Сортировать список по фамилии получателя ...............6" << endl;

cout << "Загрузить базу писем (создать при её отсутствии) .......7" << endl;

cout << "Выход из программы с сохранением изменений .............8" << endl;

cout << "Выход из программы без изменений .......................9" << endl;

cin >> p;

switch (p)

{

case 1:

add\_new\_parcel(head);

break;

case 2:

print\_all\_parcel(head);

break;

case 3:

print\_parcel\_ot(head);

break;

case 4:

filtra = filtr\_weight(head);

print\_all\_parcel(filtra);

filtra = NULL;

break;

case 5:

delete\_parcel(H);

break;

case 6:

sort\_alfavit(H);

break;

case 7:

download\_parcel(H, pth\_file);

break;

case 8:

save\_parcel(head, pth\_file);

v = false;

break;

case 9:

v = false;

break;

default:

menu(head);

v = false;

break;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

List\* HEAD = new List;

HEAD->Next = NULL;

HEAD->date.name1 = NULL;

menu(HEAD);

return 0;

}