Беларускі дзяржаўны тэхналагічны ўніверсітэт

Факультэт інфармацыйных тэхналогій

Кафедра праграмнай інжэнерыі

Лабараторная работа №9

Па дысцыпліне «Асновы алгарытмізацыі і праграмавання»

На тэму «Двухсувязные спісы»

Выканала:

Студэнтка 1 курса 6 группы

Жучкевіч Кацярына Сяргееўна

Выкладчык: асс. Андронава М.В.

Мінск, 2024



#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

struct Address

{

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

//-----------------------------------------------------------

int menu(void)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

char s[80]; int c;

cout << endl;

cout << "1. Ввод имени" << endl;

cout << "2. Удаление имени" << endl;

cout << "3. Вывод на экран" << endl;

cout << "4. Поиск" << endl;

cout << "5. Выход" << endl;

cout << "6. Пошук элемента з найменшым імем" << endl;

cout << endl;

do {

cout << "Ваш выбор: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

} while (c < 0 || c > 6);

return c;

}

//-----------------------------------------------------------

void insert(Address\* e, Address\*\* phead, Address\*\* plast) //Добавление в конец списка

{

Address\* p = \*plast;

if (\*plast == NULL)

{

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

return;

}

else

{

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

//-----------------------------------------------------------

Address\* setElement() // Создание элемента и ввод его значений с клавиатуры

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

cout << "Введите имя: ";

cin.getline(temp->name, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.clear();

cout << "Введите город: ";

cin.getline(temp->city, CITY\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

//-----------------------------------------------------------

void outputList(Address\*\* phead, Address\*\* plast) //Вывод списка на экран

{

Address\* t = \*phead;

while (t)

{

cout << t->name << ' ' << t->city << endl;

t = t->next;

}

cout << "" << endl;

}

//-----------------------------------------------------------

void find(char name[NAME\_SIZE], Address\*\* phead) // Поиск имени в списке

{

Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}if (!t)

cerr << "Имя не найдено" << endl;

else

cout << t->name << ' ' << t->city << endl;

}

//-----------------------------------------------------------

void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) // Удаление имени

{

struct Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

{

if (\*phead == t)

{

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

t->prev->next = t->next;

if (t != \*plast)

t->next->prev = t->prev;

else

\*plast = t->prev;

}

delete t; cout << "Элемент удален" << endl;

}

}

void findMin(Address\*\* phead) {

Address\* temp = \*phead;//задаем пачатковае значэнне для буфернага элемента

Address\* min = \*phead;//задаем пачатковае значэнне для мін элемента

temp = temp->next;

while (temp)

{

if (strlen(temp->name) < strlen(min->name))//шукаем мініэлемент

min = temp;

temp = temp->next;

}

cout << min->name << " " << min->city << endl;//выводзім яго

}

//-----------------------------------------------------------

int main(void)

{

Address\* head = NULL;

Address\* last = NULL;

setlocale(LC\_CTYPE, "Rus");

while (true)

{

switch (menu())

{

case 1: insert(setElement(), &head, &last);

break;

case 2: { char dname[NAME\_SIZE];

cout << "Введите имя: ";

cin.getline(dname, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

delet(dname, &head, &last);

}

break;

case 3: outputList(&head, &last);

break;

case 4: { char fname[NAME\_SIZE];

cout << "Введите имя: ";

cin.getline(fname, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

find(fname, &head);

}

break;

case 5: exit(0);

case 6: findMin(&head);

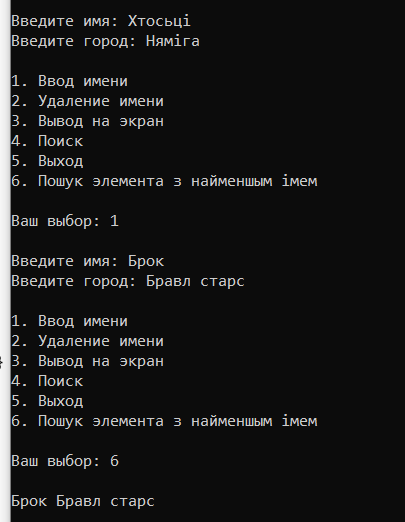
default: exit(1);

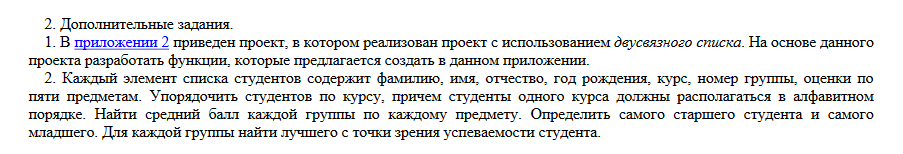
}

}

return 0;

}





**Дадатковае заданне 1**

**1.cpp**

#include "List.h"

#include <iostream>

using namespace std;

struct Person

{

char name[20];

string examName;

string examDate;

Person\* next;

};

void print(void\* b) //Функция используется при выводе

{

Person\* a = (Person\*)b;

cout << a->name << " " << a->examName << endl;

}

int main()

{

int choice;

setlocale(LC\_ALL, "Russian");

Person a1 = { "Петров", "20" };

Person a2 = { "Сидоров", "25" };

Person a3 = { "Гончаров", "47" };

bool rc;

Object L1 = Create(); // создание списка с именем L1

L1.Insert(&a1);

L1.Insert(&a2);

L1.Insert(&a3);

for (;;)

{

cout << "Увядзіце: \n1) Вывад спіса\n2) Пошук па прозвішчу\n3) Выдаліць элемент\n"

<< "4) Колькасць элементаў\n5) Ачысціць спіс\n";

cin >> choice;

switch (choice)

{

case 1: L1.PrintList(print); cout << endl; break;

case 2: L1.Search(&a2); break;

case 3: L1.Delete(&a1); cout << endl; break;

case 4: cout << L1.CountList(&L1) << "\n"; break;

case 5: L1.DeleteList(&L1); cout << endl; break;

default: exit(1);

}

}

L1.PrintList(print);

Element\* e = L1.Search(&a3);

Person\* aa = (Person\*)e->Data;

cout << "Найден элемент= " << aa->name << endl;

rc = L1.Delete(&a2);

if (rc == true) cout << "Удален элемент" << endl;

cout << "Список:" << endl;

L1.PrintList(print);

return 0;

}

**Func.cpp**

#include "List.h"

#define NULL 0

bool Object::Insert(void\* data) // Вставка в начало

{

bool rc = 0;

if (Head == NULL)

{

Head = new Element(NULL, data, Head);

rc = true;

}

else

{

Head = (Head->Prev = new Element(NULL, data, Head));

rc = true;

}

return rc;

}

//------------------ЗАДАНИЕ-----------------------------------

void Object::DeleteList(void\* data)//удаление полного списка

{

Element\* e = Head;

while (e != NULL)

{

Element\* del = e;//для хранения адреса

if (e)

{

e = e->Next;//переход на следующий элемент

}

delete del;

}

}

int Object::CountList(void\* data)//подсчет элементов списка

{

int counter = 0;

Element\* e = Head;

while (e != NULL)

{

e = e->Next;

counter++;

}

return counter;

}

//-----------------------------------------------------------

Element\* Object::Search(void\* data) // Найти заданный элемент

{

Element\* rc = Head;

while ((rc != NULL) && (rc->Data != data))

rc = rc->Next;

return rc;

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*)) // Вывод

{

Element\* e = Head;

while (e != NULL)

{

f(e->Data);

e = e->GetNext();

};

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*), Element\* e)

{

f(e->Data);

}

//-----------------------------------------------------------

bool Object::Delete(Element\* e) // Удалить по ссылке

{

bool rc = 0;

if (rc = (e != NULL))

{

if (e->Next != NULL)

e->Next->Prev = e->Prev;

if (e->Prev != NULL)

e->Prev->Next = e->Next;

else

Head = e->Next;

delete e;

}

return rc;

}

//-----------------------------------------------------------

bool Object::Delete(void\* data) // Удалить по значению

{

return Delete(Search(data));

};

//-----------------------------------------------------------

Element\* Object::GetLast()

{

Element\* e = Head, \* rc = e;

while (e != NULL)

{

rc = e;

e = e->GetNext();

}

return rc;

}

//-----------------------------------------------------------

Object Create()

{

return \*(new Object());

}

void Object::PrintList(void(\*f)(void\*)) // Вывод

{

Element\* e = Head;

while (e != NULL)

{

f(e->Data);

e = e->GetNext();

};

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*), Element\* e)

{

f(e->Data);

}

//-----------------------------------------------------------

bool Object::Delete(Element\* e) // Удалить по ссылке

{

bool rc = 0;

if (rc = (e != NULL))

{

if (e->Next != NULL)

e->Next->Prev = e->Prev;

if (e->Prev != NULL)

e->Prev->Next = e->Next;

}

return rc;

}

//-----------------------------------------------------------

Element\* Object::Search(void\* data) // Найти заданный элемент

{

Element\* rc = Head;

while ((rc != NULL) && (rc->Data != data))

rc = rc->Next;

return rc;

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*)) // Вывод

{

Element\* e = Head;

while (e != NULL)

{

f(e->Data);

e = e->GetNext();

};

}

//-----------------------------------------------------------

void Object::PrintList(void(\*f)(void\*), Element\* e)

{

f(e->Data);

}

//-----------------------------------------------------------

bool Object::Delete(Element\* e) // Удалить по ссылке

{

bool rc = 0;

if (rc = (e != NULL))

{

if (e->Next != NULL)

e->Next->Prev = e->Prev;

if (e->Prev != NULL)

e->Prev->Next = e->Next;

else

Head = e->Next;

delete e;

}

return rc;

}

//-----------------------------------------------------------

bool Object::Delete(void\* data) // Удалить по значению

{

return Delete(Search(data));

};

//-----------------------------------------------------------

Element\* Object::GetLast()

{

Element\* e = Head, \* rc = e;

while (e != NULL)

{

rc = e;

e = e->GetNext();

}

return rc;

}

//-----------------------------------------------------------

Object Create()

{

return \*(new Object());

}

void Object::DeleteList(void\* data) {//функцыя для таго, каб выдаліць спіс

Element\* start = Head;

while (start != NULL)

{

Element\* del = start;//задаем пераменнай значэнне адраса

if (start)

{

start = start->Next;//пераходзім на наступны элемент

}

delete del;//выдаляем з памяці

}

}

int Object::CountList(void\* data)//функцыя для таго, каб палічыць колькасць элементаў у спісе

{

int count = 0;

Element\* start = Head;//для таго, каб захаваць адрас

while (start != NULL)

{

count++;//лічым

start = start->Next;//пераходзім на наступны элемент

}

return count;

}

**Дадатковае заданне 2**

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

const unsigned int NAMEMAX = 30;

const unsigned int CITYSIZE = 20;

struct Student

{

char name[NAMEMAX];

char surname[CITYSIZE];

int BDayYear;

string course;

int group;

int marks[5];

Student\* next;

Student\* prev;

};

int menu() {

char s[80]; int c;

cout << endl;

cout << "1. Увод інфармацыі" << endl;

cout << "2. Выдаленне інфармацыі" << endl;

cout << "3. Вывад" << endl;

cout << "4. Пошук" << endl;

cout << "5. Выхад" << endl;

cout << "6. Сярэдні бал" << endl;

cout << "7. Найстарэйшы і наймалодшы студэнт" << endl;

cout << "8. Знайсці лепшага студэнта" << endl;

cout << endl;

do

{

cout << "Увядзіце: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

} while (c < 0 || c > 8);

return c;

}

void countAver(Student\* phead) {//функцыя для пошуку сярэднега бала

Student\* current = phead;

while (current != NULL) {

int groupNum = current->group;

double sum[5] = { 0 };

double count[5] = { 0 };

while (current != NULL && current->group == groupNum) {

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

sum[i] += current->marks[i];

++count[i];

}

current = current->next;

}

cout << "Сярэдні бал групы #" << groupNum << ":\n";

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

double average = (sum[i]) / count[i];

cout << "Прадмет " << i + 1 << " - " << average << "\n";

}

cout << "\n";

}

}

void findYunNOld(Student\* phead) { //функцыя для пошуку па году нараджэння

Student\* youngest = phead;

Student\* oldest = phead;

int findYun = 2024 - youngest->BDayYear;

int findOld = 2024 - oldest->BDayYear;

Student\* finder = phead->next;

while (finder != NULL) {//пакуль не канчатак

int temp = 2024 - finder->BDayYear;

if (temp < findYun)

{

findYun = temp;

youngest = finder;

}

else if (temp > findOld)

{

findOld = temp;

oldest = finder;

}

finder = finder->next;

}

cout << "Найстарэйшы: " << oldest->surname << " " << oldest->name << "\n";

cout << "Наймалодшы: " << youngest->surname << " " << youngest->name << "\n";

}

void findBestStudent(Student\* phead)

{

setlocale(LC\_CTYPE, "Rus");

Student\* find = phead;

while (find != NULL)

{

int groupNum = find->group;

Student\* theBest = find;

double temp = 0;

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

{

temp += theBest->marks[i];

}

double bestAverage = temp / 5;

while (find != NULL and find->group == groupNum)

{

double temp1 = 0;

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

{

temp1 += find->marks[i];

}

double average = temp1 / 5;

if (average > bestAverage)

{

bestAverage = average;

theBest = find;

}

find = find->next;

}

cout << "Лепшы студэнт - " << groupNum << theBest->surname << " "

<< theBest->name << "\n";

}

}

void insert(Student\* e, Student\*\* phead, Student\*\* plast) //Добавление в конец списка

{

Student\* p = \*plast;

if (\*plast == NULL)

{

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

return;

}

else

{

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

Student\* setElement() // Создание элемента и ввод его значений с клавиатуры

{

Student\* temp = new Student();

if (!temp)

{

cerr << "Памылка!!";

return NULL;

}

cout << "Увядзіце імя: ";

cin >> temp->name;

cin.ignore();

cout << "Увядзіце прозвішча: ";

cin >> temp->surname;

cin.ignore();

cout << "Увядзіце год нараджэння: ";

cin >> temp->BDayYear;

cin.ignore();

cout << "Увядзіце курс: ";

getline(cin, temp->course);

cout << "Увядзіце групу: ";

cin >> temp->group;

cin.ignore();

cout << "Увядзіце 5 адзнак: ";

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

cin >> temp->marks[i];

}

temp->next = NULL;

temp->prev = NULL;

cin.ignore();

return temp;

}

//-----------------------------------------------------------

void outputList(Student\*\* phead, Student\*\* plast) //Вывод списка на экран

{

Student\* t = \*phead;

while (t)

{

cout << t->name << ' ' << t->surname << endl;

t = t->next;

}

cout << "" << endl;

}

//-----------------------------------------------------------

void find(char name[NAMEMAX], Student\*\* phead) // Поиск имени в списке

{

Student\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

cout << t->name << ' ' << t->surname << endl;

}

//-----------------------------------------------------------

void delet(char name[NAMEMAX], Student\*\* phead, Student\*\* plast) // Удаление имени

{

struct Student\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

{

if (\*phead == t)

{

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

t->prev->next = t->next;

if (t != \*plast)

t->next->prev = t->prev;

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl;

}

}

int main() {

Student\* head = NULL;

Student\* last = NULL;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (true)

{

switch (menu())

{

case 1: insert(setElement(), &head, &last);

break;

case 2: { char dname[NAMEMAX];

cout << "Введите имя: ";

cin.getline(dname, NAMEMAX - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

delet(dname, &head, &last);

}

break;

case 3: outputList(&head, &last);

break;

case 4: { char fname[NAMEMAX];

cout << "Увядзіце імя: ";

cin.getline(fname, NAMEMAX - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

find(fname, &head);

}

break;

case 5: exit(0);

case 6: countAver(head); break;

case 7: findYunNOld(head); break;

case 8: findBestStudent(head); break;

default: exit(1);

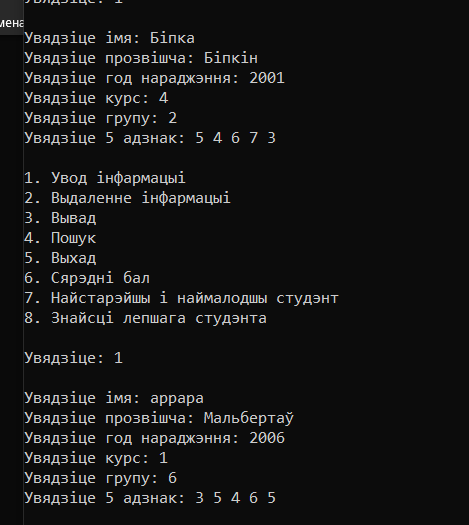
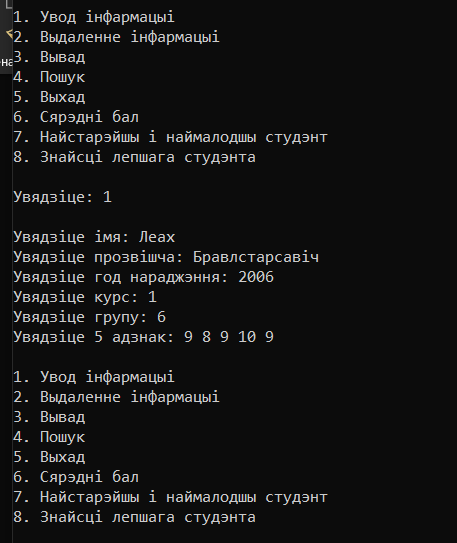
}

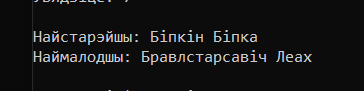
}

return 0;

}

**//каментарыі не змяняла, бо гэта з метадычкі**

****

****