Лабораторная работа 9

По дисциплине «Основы алгоритмизации и программирования»

На тему «Двусвязные списки»

***Задание#1***



*Macros.h*

*.cpp*

#include "Macros.h"

const unsigned int NAME\_SIZE = 30;

const unsigned int SURNAME\_SIZE = 20;

struct name\_surname

{

char name[NAME\_SIZE];

char surname[SURNAME\_SIZE];

name\_surname\* next;

name\_surname\* prev;

};

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

int menu(void)

{

char s[80]; int c;

nl

vi "1. Ввод имени"; nl

vi "2. Удаление имени"; nl

vi "3. Вывод на экран"; nl

vi "4. Поиск"; nl

vi "5. Удаление M-ого"; nl

vi "6. Выход"; nl

nl

do {

vi "Ваш выбор: ";

cin.sync();

gets\_s(s);

nl

c = atoi(s);

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

ret c;

}

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

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

{

name\_surname\* 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;

}

}

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

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

{

name\_surname\* temp = new name\_surname();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

vi "Введите имя: ";

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

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

cin.clear();

vi "Введите фамилию: ";

cin.getline(temp->surname, SURNAME\_SIZE - 1, '\n');

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

ret temp;

}

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

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

{

name\_surname\* t = \*phead;

while (t)

{

vi t->name << ' ' << t->surname; nl

t = t->next;

}

nl

}

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

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

{

name\_surname\* t = \*phead;

while (t)

{

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

t = t->next;

}

if (!t)

{

cerr << "Имя не найдено"; nl

}

else

{

vi t->name << ' ' << t->surname; nl

}

}

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

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

{

struct name\_surname\* t = \*phead;

while (t)

{

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

t = t->next;

}

if (!t)

{

cerr << "Имя не найдено"; nl

}

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;

vi "Элемент удален"; nl

}

}

// Функция для удаления каждого M-ого элемента списка

void deleteEveryM(name\_surname\*\* phead, int m)

{

if (m <= 0)

{

cerr << "Некорректное значение M"; nl

ret;

}

name\_surname\* current = \*phead;

int count = 1;

while (current != nullptr)

{

if (count % m == 0)

{

name\_surname\* toDelete = current;

current = current->next; // к следующему элементу перед удалением

delet(toDelete->name, phead, nullptr);

}

else

{

current = current->next;

}

count++;

}

}

int main(void) {

name\_surname\* head = nullptr;

name\_surname\* last = nullptr;

setlocale(LC\_CTYPE, "Rus"); // Установка локали

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];

vi "Введите имя: ";

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

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

cin.sync();

find(fname, &head);

}

break;

case 5: {

int m = -1;

vi "Введите M: ";

vv m;

deleteEveryM(&head, m);

outputList(&head, &last);

}

break;

case 6: exit(0); break;

default: exit(1);

}

}

pause

ret 0;

}

