Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 8

По дисциплине «Основы алгоритмизации и программирования»

На тему «Полустатические структуры данных: очереди»

Выполнил:

Студент 1 курса 8 группы

Статько Герман Вячеславович

Преподаватель: асс. Андронова М.В.

Минск, 2024

Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 8

По дисциплине «Основы алгоритмизации и программирования»

На тему «Полустатические структуры данных: очереди»

Выполнил:

Студент 1 курса 8 группы

Статько Герман Вячеславович

Преподаватель: асс. Андронова М.В.

Минск, 2024

**Дополнить программу функцией в соответствии со своим вариантом из таблицы, представленной ниже.**

**Вариант 12**

****

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

// Структура для элемента списка

struct Address

{

int age;

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

// Функция для создания нового элемента списка

Address\* setElement();

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast);

// Функция для вывода списка на экран

void outputList(Address\*\* phead, Address\*\* plast);

// Функция для удаления элемента по имени

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

// Функция для поиска элемента по имени

void find(char name[NAME\_SIZE], Address\*\* phead);

// Функция для удаления первого встречающегося элемента с заданным возрастом

void deleteX(Address\*\* phead, Address\*\* plast, int x);

int main()

{

// Установка кодировки для корректного ввода и вывода на кириллице в консоли Windows

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "rus");

Address\* head = NULL;

Address\* last = NULL;

char choice;

// Вывод меню пользователю

cout << "1 - Добавление информации" << endl

<< "2 - Удаление информации" << endl

<< "3 - Вывод всех элементов" << endl

<< "4 - Поиск" << endl

<< "5 - Удаление первого встречающегося элемента с заданным возрастом" << endl

<< "6 - Выход" << endl;

do

{

cout << "?: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case '1':

// Добавление нового элемента в конец списка

insert(setElement(), &head, &last);

break;

case '2':

// Удаление элемента по имени

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':

// Вывод всех элементов списка

outputList(&head, &last);

break;

case '4':

// Поиск элемента по имени

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':

// Удаление первого встречающегося элемента с заданным возрастом

int x;

cout << "Введите возраст: ";

cin >> x;

deleteX(&head, &last, x);

break;

case '6':

cout << "Выход..." << endl;

break;

default:

cout << "Неправильный выбор" << endl;

break;

}

} while (choice != '6');

}

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast)

{

Address\* p = \*plast;

if (\*plast == NULL)

{

// Если список пуст, новый элемент становится началом и концом списка

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

}

else

{

// Иначе добавляем новый элемент в конец списка

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

// Функция для создания нового элемента списка

Address\* setElement()

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти";

return NULL;

}

// Ввод данных для нового элемента

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();

cout << "Введите возраст: ";

cin >> temp->age;

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

// Функция для вывода списка на экран

void outputList(Address\*\* phead, Address\*\* plast)

{

Address\* t = \*phead;

while (t)

{

// Вывод информации об элементе списка

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

t = t->next;

}

cout << endl;

}

// Функция для удаления элемента по имени

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

{

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)

{

// Если удаляемый элемент - начало списка

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

// Иначе удаляем элемент из середины или конца списка

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

if (t != \*plast)

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

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl << endl;

}

}

// Функция для поиска элемента по имени

void find(char name[NAME\_SIZE], Address\*\* phead)

{

Address\* t = \*phead;

while (t)

{

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

break;

t = t->next;

}

if (!t)

{

cerr << "Имя не найдено" << endl;

}

else

{

// Вывод найденного элемента

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

}

}

// Функция для удаления первого встречающегося элемента с заданным возрастом

void deleteX(Address\*\* phead, Address\*\* plast, int x)

{

Address\* t = \*phead;

while (t)

{

if (t->age == x)

break;

t = t->next;

}

if (!t)

{

cerr << "Человек с таким возрастом не найден" << endl;

}

else

{

if (\*phead == t)

{

// Если удаляемый элемент - начало списка

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

// Иначе удаляем элемент из середины или конца списка

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

if (t != \*plast)

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

else

\*plast = t->prev;

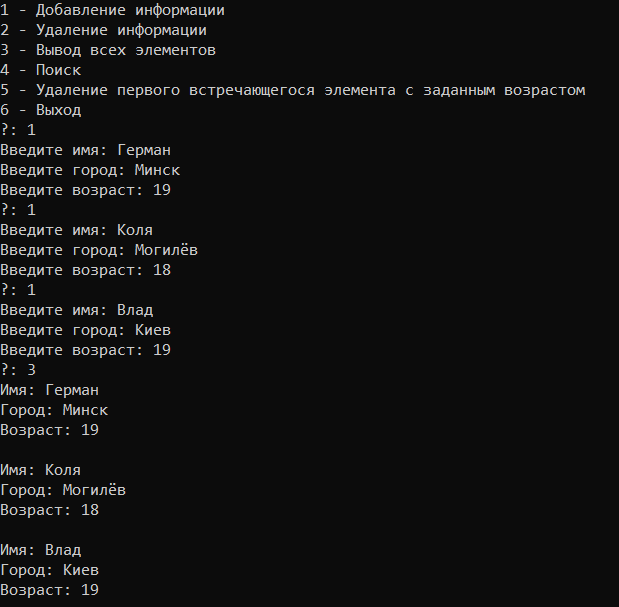
}

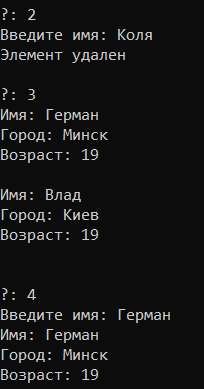
delete t;

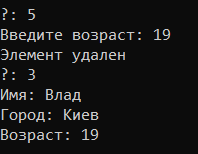
cout << "Элемент удален" << endl;

}

}

****

****

****

**Доп. Задания**

**Вариант 16**

****

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

// Структура для элемента списка

struct Address

{

int age;

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

// Функция для создания нового элемента списка

Address\* setElement();

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast);

// Функция для вывода списка на экран

void outputList(Address\*\* phead, Address\*\* plast);

// Функция для удаления элемента по имени

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

// Функция для поиска элемента по имени

void find(char name[NAME\_SIZE], Address\*\* phead);

// Функция для удаления первого встречающегося элемента с заданным возрастом

void countX(Address\*\* phead, Address\*\* plast, int x);

int main()

{

// Установка кодировки для корректного ввода и вывода на кириллице в консоли Windows

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_CTYPE, "rus");

Address\* head = NULL;

Address\* last = NULL;

char choice;

// Вывод меню пользователю

cout << "1 - Добавление информации" << endl

<< "2 - Удаление информации" << endl

<< "3 - Вывод всех элементов" << endl

<< "4 - Поиск" << endl

<< "5 - Подсчет элементов с заданным возрастом" << endl

<< "6 - Выход" << endl;

do

{

cout << "?: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case '1':

// Добавление нового элемента в конец списка

insert(setElement(), &head, &last);

break;

case '2':

// Удаление элемента по имени

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':

// Вывод всех элементов списка

outputList(&head, &last);

break;

case '4':

// Поиск элемента по имени

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':

// Удаление первого встречающегося элемента с заданным возрастом

int x;

cout << "Введите возраст: ";

cin >> x;

countX(&head, &last, x);

break;

case '6':

cout << "Выход..." << endl;

break;

default:

cout << "Неправильный выбор" << endl;

break;

}

} while (choice != '6');

}

// Функция для вставки элемента в конец списка

void insert(Address\* e, Address\*\* phead, Address\*\* plast)

{

Address\* p = \*plast;

if (\*plast == NULL)

{

// Если список пуст, новый элемент становится началом и концом списка

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

}

else

{

// Иначе добавляем новый элемент в конец списка

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

// Функция для создания нового элемента списка

Address\* setElement()

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти";

return NULL;

}

// Ввод данных для нового элемента

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();

cout << "Введите возраст: ";

cin >> temp->age;

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

cin.clear();

temp->next = NULL;

temp->prev = NULL;

return temp;

}

// Функция для вывода списка на экран

void outputList(Address\*\* phead, Address\*\* plast)

{

Address\* t = \*phead;

while (t)

{

// Вывод информации об элементе списка

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

t = t->next;

}

cout << endl;

}

// Функция для удаления элемента по имени

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

{

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)

{

// Если удаляемый элемент - начало списка

\*phead = t->next;

if (\*phead)

(\*phead)->prev = NULL;

else

\*plast = NULL;

}

else

{

// Иначе удаляем элемент из середины или конца списка

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

if (t != \*plast)

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

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl << endl;

}

}

// Функция для поиска элемента по имени

void find(char name[NAME\_SIZE], Address\*\* phead)

{

Address\* t = \*phead;

while (t)

{

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

break;

t = t->next;

}

if (!t)

{

cerr << "Имя не найдено" << endl;

}

else

{

// Вывод найденного элемента

cout << "Имя: " << t->name << endl

<< "Город: " << t->city << endl

<< "Возраст: " << t->age << endl << endl;

}

}

// Функция для подсчета всех элементов с заданным значением x

void countX(Address\*\* phead, Address\*\* plast, int x)

{

int count = 0;

Address\* t = \*phead;

while (t)

{

if (t->age == x)

count++;

t = t->next;

}

if (count == 0)

{

cout << "Людей с таким возрастом не найдено" << endl << endl;

}

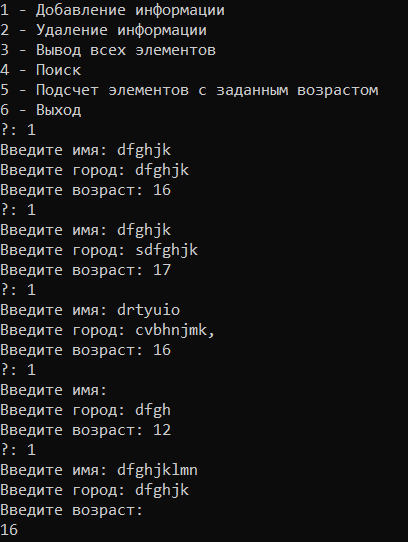
else

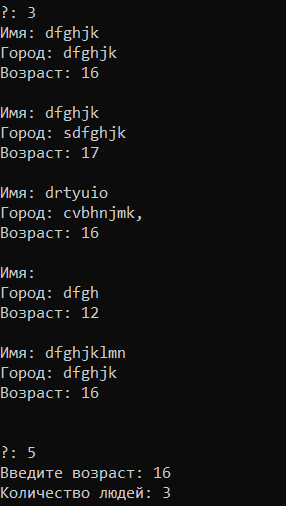
{

cout << "Количество людей: " << count << endl << endl;

}

}

****

****