**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Кафедра «Вычислительная техника»**

**ОТЧЁТ**

**по дисциплине «Л и ОА в ИЗ»**

**по лабораторной работе №3 «динамические списки»**

Выполнил студент гр. 22ВВВ3:

Андреянов Я.И.

Приняли:

к.т.н., доцент Юрова О.В.

к.э.н., доцент Акифьев И.В.

Пенза

2023

**Цель:**

В данной лабораторной работе мы изучаем принципы работы стека, очередь и очередь с приоритетом на языке программирования C++.

**Общие сведения:**

Список представляет собой последовательность элементов определенного типа. Простейший тип списка – линейный, когда для каждого из элементов, кроме последнего, имеется следующий, и для каждого, кроме первого имеется предыдущий элемент.

Возможна реализация списков посредством массивов или динамическая реализация.

Динамические списки относятся к динамическим структурам и используются, когда размер данных заранее неизвестен. Созданием динамических данных должна заниматься сама программа во время своего исполнения, этим достигается эффективное распределение памяти, но снижается эффективность доступа к элементам.

Динамические структуры данных отличаются от статических двумя основными свойствами:

1. В них нельзя обеспечить хранение в заголовке всей информации о структуре, поэтому каждый элемент должен содержать информацию, логически связывающую его с другими элементами структуры;

2. Для них зачастую не удобно использовать единый массив смежных элементов памяти, поэтому необходимо предусматривать ту или иную схему динамического управления памятью.

Для обращения к динамическим данным применяют указатели.

Набор операций над списком будет включать добавление и удаление элементов, поиск элементов списка.

Различают односвязные, двусвязные и циклические списки.

В простейшем случае каждый элемент содержит всего одну ссылку на следующий элемент, такой список называется односвязным.

В простейшем случае для создания элемента списка используется структура, в которой объединяются полезная информация и ссылка на следующий элемент списка.

Обращение к списку и его элементам осуществляется посредством указателей.

Для списка реализованы функции создания, добавления, удаления элемента, просмотра списка, нахождения нужного элемента списка.

**Практическая часть:**

**Задание 1:**

Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).

**Задание 2:**

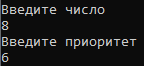
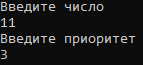
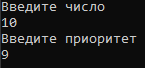
\* На основе приведенного кода реализуйте структуру данных *Очередь*.

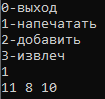
**Задание 2:**

\* На основе приведенного кода реализуйте структуру данных *Стек*.

**Расчёты:**

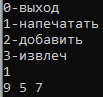
**Задание 1:**

****

****

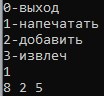
**Задание 2:**

****

****

**Задание 3:**

**** **** 

****

**Заключение:**

В данной лабораторной работе были изучены такие структуры данных как очередь, стек и очередь с приоритетом. Получены навыки работы с двусвязным циклическим списком.

**Листинг:**

**spiski.h**

#include <Windows.h>

#include <iostream>

using namespace std;

typedef struct spisok

{

int num, preoriti;

spisok\* next, \* back;

}sip;

class Spisochek

{

protected:

spisok\* start, \* end;

public:

virtual void add(int) = 0;

virtual void print() = 0;

virtual void pop() = 0;

};

class Queue : public Spisochek {

public:

Queue() { start = NULL; end = NULL; };

void add(int num) override;

public: void print() override;

void pop() override;

};

class PQueue : public Queue

{

public: void add(int num, int prioriti);

};

class Stack : public Queue

{

public:

void pop();

};

**main.cpp**

#include "spiski.h"

template <typename T>

void menu(T str);

int main()

{

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice, value, prioriti;

cout << "Выберите структуру:\n1-стек\n2-очередь\n3-приоритетная очередь" << endl;

cin >> choice;

if (choice == 1)

{

Stack st;

while (1)

{

system("clear");

cout << "Выберите действие\n0-выход\n1-напечатать\n2-добавить\n3-извлеч" << endl;

cin >> choice;

if (choice == 1) st.print();

else if (choice == 2)

{

cout << "Введите число" << endl;

cin >> value;

st.add(value);

}

else if (choice == 3) st.pop();

else if (choice == 0) break;

else cout << "Неверное число!" << endl;

}

}

else if (choice == 2)

{

Queue qe;

while (1)

{

system("clear");

cout << "Выберите действие\n0-выход\n1-напечатать\n2-добавить\n3-извлеч" << endl;

cin >> choice;

if (choice == 1) qe.print();

else if (choice == 2)

{

cout << "Введите число" << endl;

cin >> value;

qe.add(value);

}

else if (choice == 3) qe.pop();

else if (choice == 0) break;

else cout << "Неверное число!" << endl;

}

}

else if (choice == 3)

{

PQueue qe;

while (1)

{

system("clear");

cout << "Выберите действие\n0-выход\n1-напечатать\n2-добавить\n3-извлеч" << endl;

cin >> choice;

if (choice == 1) qe.print();

else if (choice == 2)

{

cout << "Введите число" << endl;

cin >> value;

cout << "Введите приоритет" << endl;

cin >> prioriti;

qe.add(value, prioriti);

}

else if (choice == 3) qe.pop();

else if (choice == 0) break;

else cout << "Неверное число!" << endl;

}

}

else cout << "Неверное число!" << endl;

return 0;

}

**spiski.cpp**

#include "spiski.h"

void Queue::add(int num)

{

spisok\* tmp;

if (!start)

{

tmp = new spisok;

tmp->num = num;

tmp->next = NULL;

tmp->back = NULL;

this->start = tmp;

}

else

{

spisok\* sps = start;

while (sps->next != NULL)

{

sps = sps->next;

}

tmp = new spisok;

tmp->num = num;

tmp->next = NULL;

tmp->back = sps;

sps->next = tmp;

end = tmp;

}

}

void Queue::print()

{

spisok\* sps = this->start;

while (1)

{

cout << sps->num << " ";

if (!sps->next) break;

sps = sps->next;

}

cout << endl;

cin.ignore();

cin.get();

}

void Queue::pop()

{

spisok\* tmp;

if (!start)

{

cout << "Список пуст!" << endl;

return;

}

tmp = start;

if (start->next == NULL) start = NULL;

else

{

start = start->next;

start->back = NULL;

}

cout << tmp->num << endl;

free(tmp);

cin.ignore();

cin.get();

}

void Stack::pop()

{

spisok\* tmp;

if (!start)

{

cout << "Список пуст!" << endl;

}

tmp = end;

if (end->back == NULL) start = NULL;

else

{

end = end->back;

end->next = NULL;

}

cout << tmp->num << endl;

free(tmp);

cin.ignore();

cin.get();

};

void PQueue::add(int num, int preoriti)

{

spisok\* tmp, \* sps;

if (!start)

{

tmp = new spisok;

tmp->num = num;

tmp->preoriti = preoriti;

tmp->next = NULL;

tmp->back = NULL;

this->start = tmp;

this->end = tmp;

}

else

{

sps = start;

tmp = new spisok;

tmp->num = num;

tmp->preoriti = preoriti;

while (1)

{

if (preoriti < sps->preoriti)

{

spisok\* pred = sps->back;

tmp->next = sps;

if (!pred)

{

this->start = tmp;

tmp->back = NULL;

}

else

{

tmp->back = pred;

pred->next = tmp;

sps->back = tmp;

}

break;

}

else if (sps->next == NULL)

{

tmp->next = NULL;

tmp->back = sps;

sps->next = tmp;

end = tmp;

break;

}

sps = sps->next;

}

}

};