Министерство науки и высшего образования

Пензенский государственный университет

Кафедра «Вычислительная техника»

Отчет

По лабораторной работе №3

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Динамические списки»

Выполнили студенты группы 21ВВ2:

Алёшина А.В.

Мубаряков А.А.

Приняли:

Юрова О.В.

Митрохин М.А.

Пенза 2022

Ход работы:

Header.h

Здесь хранятся все библиотеки.

#pragmaonce

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

#include <iostream>

#include <windows.h>

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

#include "ElementLists.h"

#include "PriorityQueue.h"

#include "Queue.h"

#include "Stack.h"

main.cpp

Содержит взаимодействие всех трёх элементов PresentationElementLists, PresentationPriorityQueue, PresentationQueue.

#include "Header.h"

int PresentationPriorityQueue()

{

try

{

PriorityQueue priorityQueue;

int value, priority;

std::cout << "Введите значение элемента: ";

std::cin >> value;

std::cout << "Введите приоритет элемента: ";

std::cin >> priority;

priorityQueue.push(value, priority);

priorityQueue.output();

std::cout << std::endl;

std::cout << "Введите значение элемента: ";

std::cin >> value;

std::cout << "Введите приоритет элемента: ";

std::cin >> priority;

priorityQueue.push(value, priority);

priorityQueue.output();

std::cout << std::endl;

std::cout << "Введите значение элемента: ";

std::cin >> value;

std::cout << "Введите приоритет элемента: ";

std::cin >> priority;

priorityQueue.push(value, priority);

priorityQueue.output();

std::cout << std::endl;

std::cout << "Полученные значения: " << std::endl;

priorityQueue.pop();

priorityQueue.output();

std::cout << std::endl;

priorityQueue.pop();

priorityQueue.output();

std::cout << std::endl;

priorityQueue.pop();

priorityQueue.output();

std::cout << std::endl;

}

catch (...)

{

std::cerr << "Возникла непредвиденная ошибка. Необходимо запустить средства отладки." << std::endl;

return 1;

}

return 0;

}

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

int PresentationQueue()

{

try

{

Queue queue;

queue.push(1);

queue.push(2);

queue.output();

std::cout << std::endl;

queue.pop();

queue.output();

std::cout << std::endl;

queue.pop();

queue.output();

std::cout << std::endl;

}

catch (...)

{

std::cerr << "Возникла непредвиденная ошибка. Необходимо запустить средства отладки." << std::endl;

return 1;

}

return 0;

}

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

int PresentationStack()

{

try

{

Stack stack;

stack.push(1);

stack.push(2);

stack.output();

std::cout << std::endl;

stack.pop();

stack.output();

std::cout << std::endl;

stack.pop();

stack.output();

std::cout << std::endl;

}

catch (...)

{

std::cerr << "Возникла непредвиденная ошибка. Необходимо запустить средства отладки." << std::endl;

return 1;

}

return 0;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

PresentationPriorityQueue();

PresentationQueue();

PresentationStack();

return 0;

}

Библиотека ElementLists.h

ElementLists.h – это библиотека, которая подключает единицу

элемента

#pragma once

class ElementLists

{

int m\_value = 0; //Значение

int m\_number = 0; //Номер элемента

int m\_prioritet = 0; //Приоритет элемента

public:

ElementLists\* next = nullptr;

int getValue();

int getNumber();

int getPrioritet();

void setValue(int \_value);

void setNumber(int \_number);

void setPrioritet(int \_prioritet);

void output(int number);

};

ElementLists.cpp

#include "Header.h"

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

int ElementLists::getValue()

{

return m\_value;

}

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

int ElementLists::getNumber()

{

return m\_number;

}

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

int ElementLists::getPrioritet()

{

return m\_prioritet;

}

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

void ElementLists::setValue(int \_value)

{

m\_value = \_value;

}

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

void ElementLists::setNumber(int \_number)

{

m\_number = \_number;

}

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

void ElementLists::setPrioritet(int \_prioritet)

{

m\_prioritet = \_prioritet;

}

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

void ElementLists::output(int number)

{

std::cout << "Элемент №" << number << std::endl;

std::cout << "Приоритет элемента: " << m\_prioritet << std::endl;

std::cout << "Значение элемента: " << m\_value << std::endl;

}

Библиотека PriorityQueue.h

PriorityQueue.h – это библиотека для приоритетной очереди.

#pragma once

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

class PriorityQueue

{

ElementLists\* m\_first = nullptr;

ElementLists\* m\_last = nullptr;

int m\_qualityElements = 0;

public:

bool empty();

int push(int \_value, int \_prioritet);

int pop();

void output();

};

PriorityQueue.cpp

#include "Header.h"

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

bool PriorityQueue::empty()

{

return m\_first == nullptr;

}

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

int PriorityQueue::push(int \_value, int \_prioritet)

{

if (empty())

{

m\_first = new ElementLists;

m\_last = m\_first;

m\_first->setValue(\_value);

++m\_qualityElements;

m\_first->setNumber(m\_qualityElements);

m\_first->setPrioritet(\_prioritet);

return 0;

}

for (ElementLists\* pointer = m\_first; pointer != nullptr; pointer = pointer->next)

{

if (pointer->getPrioritet() == \_prioritet) return 1;

}

if (m\_first->getPrioritet() < \_prioritet)

{

ElementLists\* oldFirst = m\_first;

m\_first = new ElementLists;

m\_first->next = oldFirst;

m\_first->setValue(\_value);

m\_first->setPrioritet(\_prioritet);

++m\_qualityElements;

//Увелечение порядкого номера всех элементов

int counterNumber = 0;

for (ElementLists\* pointer = m\_first; pointer != nullptr; pointer = pointer->next)

{

++counterNumber;

pointer->setNumber(counterNumber);

}

return 0;

}

ElementLists\* pointerBefore = m\_first;

for (pointerBefore = m\_first; pointerBefore->next != nullptr && pointerBefore->next->getPrioritet() > \_prioritet; pointerBefore = pointerBefore->next);

ElementLists\* pointerAfter = pointerBefore->next;

ElementLists\* pointer = new ElementLists;

pointerBefore->next = pointer;

pointer->next = pointerAfter;

pointer->setValue(\_value);

pointer->setNumber(pointerBefore->getNumber() + 1);

pointer->setPrioritet(\_prioritet);

++m\_qualityElements;

int counterNumber = pointer->getNumber();

for (ElementLists\* \_pointer = pointer->next; \_pointer != nullptr; \_pointer = \_pointer->next)

{

++counterNumber;

\_pointer->setNumber(counterNumber);

}

return 0;

}

int PriorityQueue::pop()

{

int \_value = m\_first->getValue();

--m\_qualityElements;

ElementLists\* newFirst = m\_first->next;

delete m\_first;

m\_first = newFirst;

return \_value;

}

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

void PriorityQueue::output()

{

if (empty())

{

std::cout << "Приоритетная очередь не содержит элементов." << std::endl;

return;

}

std::cout << "Приоритетная очередь содержит: " << std::endl;

int counter = 1;

for (ElementLists\* pointerEL = m\_first; pointerEL != nullptr; pointerEL = pointerEL->next, ++counter)

{

pointerEL->output(counter);

}

}

Библиотека Queue.h

Queue.h – это библиотека для очереди.

#pragma once

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

class Queue

{

ElementLists\* m\_first = nullptr;

ElementLists\* m\_last = nullptr;

int m\_qualityElements = 0;

public:

bool empty();

int push(int \_value);

int pop();

void output();

};

Queue.cpp

#include "Header.h"

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

bool Queue::empty()

{

return m\_first == nullptr;

}

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

int Queue::push(int \_value)

{

if (empty())

{

m\_first = new ElementLists;

m\_last = m\_first;

m\_first->setValue(\_value);

++m\_qualityElements;

m\_first->setNumber(m\_qualityElements);

m\_first->setPrioritet(0);

return 0;

}

++m\_qualityElements;

m\_last->next = new ElementLists;

m\_last = m\_last->next;

m\_last->setNumber(m\_qualityElements);

m\_last->setPrioritet(0);

m\_last->setValue(\_value);

return 0;

}

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

int Queue::pop()

{

int \_value = m\_first->getValue();

--m\_qualityElements;

ElementLists\* newFirst = m\_first->next;

delete m\_first;

m\_first = newFirst;

return \_value;

}

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

void Queue::output()

{

if (empty())

{

std::cout << "Очередь не содержит элементов." << std::endl;

return;

}

std::cout << "Очередь содержит: " << std::endl;

int counter = 1;

for (ElementLists\* pointerEL = m\_first; pointerEL != nullptr; pointerEL = pointerEL->next, ++counter)

{

pointerEL->output(counter);

}

}

БиблиотекаStack.h

Stack.h – это библиотека для стека.

#pragma once

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

class Stack

{

ElementLists\* m\_first = nullptr;

ElementLists\* m\_last = nullptr;

int m\_qualityElements = 0;

public:

bool empty();

int push(int \_value);

int pop();

void output();

};

Stack.cpp

#include "Header.h"

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

bool Stack::empty()

{

return m\_first == nullptr;

}

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

int Stack::push(int \_value)

{

if (empty())

{

m\_first = new ElementLists;

m\_last = m\_first;

m\_first->setValue(\_value);

++m\_qualityElements;

m\_first->setNumber(m\_qualityElements);

m\_first->setPrioritet(0);

return 0;

}

++m\_qualityElements;

m\_last->next = new ElementLists;

m\_last = m\_last->next;

m\_last->setNumber(m\_qualityElements);

m\_last->setPrioritet(0);

m\_last->setValue(\_value);

return 0;

}

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

int Stack::pop()

{

if (empty())return -0; //maybe using throw?)

int \_value = m\_last->getValue();

if (m\_qualityElements == 1)

{

--m\_qualityElements;

delete m\_first;

m\_first = nullptr;

m\_last = nullptr;

return \_value;

}

--m\_qualityElements;

ElementLists\* newLast = m\_first;

for (; newLast->next != m\_last; newLast = newLast->next);

delete m\_last;

m\_last = newLast;

m\_last->next = nullptr;

return \_value;

}

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

void Stack::output()

{

if (empty())

{

std::cout << "Стек не содержит элементов." << std::endl;

return;

}

std::cout << "Стек содержит: " << std::endl;

int counter = 1;

for (ElementLists\* pointerEL = m\_first; pointerEL != nullptr; pointerEL = pointerEL->next, ++counter)

{

pointerEL->output(counter);

}

}

Вывод: реализовали стек, очередь и приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта.