Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт: Институт цифрового развития

Кафедра: Межинститутская базовая кафедра

**Отчёт по Лабораторной работе № 1 по информационным технологиям и программированию**

выполнил:

Студент, 1 курс, группа ПИЖ-б-о-23-2

направление подготовки

09.03.04 Программная инженерия,

направленность «Разработка и сопровождение программного обеспечения »

форма обучения очная

Черников Дмитрий Дмитриевич

Ставрополь, 2024г

**Лабораторная работа № 1. Динамические структуры: стеки, очереди и списки**

**Цель работы**: освоить программирование задач с динамическими структурами (стеки, очереди, списки)

**Вариант** – 9

GitHub: https://github.com/dimain1/information\_technology\_and\_programming/tree/main/Lab1

**Задание 1**: Написать программу в соответствии с индивидуальным заданием.

1) Создать очередь из целых чисел. Определить количество положительных элементов очереди. Организовать просмотр данных очереди.

код программы:

void Task1() {

int positiveNums = 0;

int n;

cout << "Введите длину очереди ";

cin >> n;

queue<int> queueOfIntegers;

string next;

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

cin >> next;

if (stoi(next) >= 0) {

positiveNums++;

}

queueOfIntegers.push(stoi(next));

}

while (queueOfIntegers.empty() != true) {

cout << queueOfIntegers.front() << " ";

queueOfIntegers.pop();

}

cout << "\n" << "Количество положительных чисел = " << positiveNums;

}

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

2) Создать стек, информационными полями которого являются: название горы и высота. Добавить в стек сведения о новой горе. Организовать просмотр данных стека и определить среднюю высоту гор.

Код программы:

void Task2() {

struct Mountain {

std::string name;

int height;

};

int count = 0;

std::stack<Mountain> mountainStack;

cout << "Введите количество гор ";

cin >> count;

string moutainName = "";

int mountainHeight = 0;

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

cout << "Введите название горы ";

cin >> moutainName;

cout << "Введите высоту горы ";

cin >> mountainHeight;

mountainStack.push({ moutainName, mountainHeight });

}

int totalHeight = 0;

while (!mountainStack.empty()) {

Mountain curMountain = mountainStack.top();

totalHeight += curMountain.height;

std::cout << "Гора: " << curMountain.name << ", Высота: " << curMountain.height << std::endl;

mountainStack.pop();

}

if (count > 0) {

float averHeight = static\_cast<float>(totalHeight) / count;

std::cout << "Средняя Высота: " << averHeight << std::endl;

}

}

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

**Задание 2**: написать программу в соответствии с индивидуальным заданием

1) Создать очередь строковых значений, для реализации используя одно- связные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four» и распечатайте содержимое очереди. Удалите 1 элемент из очереди, затем добавьте в очередь строку «five» и распечатайте очередь еще раз. Найдите суммарную длину всех строк, принадлежащих очереди.

Код программы:

#include<string>

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

struct Element {

string value;

Element\* next;

Element(string data) : value(data), next(nullptr){}

};

void enqueue(Element\* &start, Element\* &end, string value) {

Element\* newElement = new Element(value);

if (start == nullptr) {

start = newElement;

end = newElement;

}

else {

end->next = newElement;

end = newElement;

}

}

void printQueue(Element\* start) {

Element now = \*start;

if (start == nullptr) {

cout << "Очередь пуста";

}

else {

while(now.next != nullptr) {

cout << now.value << " ";

now = \*now.next;

}

cout << now.value << endl;

}

}

void dequeue(Element\* &start, Element\* &end) {

if (start == nullptr) {

cout << "Очередь пуста";

}

else {

Element\* temp = start;

start = start->next;

delete temp;

}

}

int main()

{

setlocale(LC\_ALL, "ru");

Element\* start = nullptr;

Element\* end = nullptr;

enqueue(start, end, "one");

enqueue(start, end, "two");

enqueue(start, end, "three");

enqueue(start, end, "four");

printQueue(start);

dequeue(start,end);

enqueue(start, end, "five");

printQueue(start);

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

2) Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа –2.1, 1.3, -1.34, 3.3 и распечатайте содержимое очереди. Удалите 1 элемент из очереди, затем добавьте в очередь число 2.9 и распечатайте очередь еще раз. Найдите сумму отрицательных элементов очереди.

Код программы:

#include<iostream>

using namespace std;

struct Element {

double data;

Element\* next;

Element(double value) : data(value), next(nullptr) {}

};

void enqueue(Element\* &start, Element\* &end, double value ) {

Element\* newElement = new Element(value);

if (start == nullptr) {

start = end = newElement;

}

else {

end->next = newElement;

end = newElement;

}

}

void dequeue(Element\* &start) {

if (start == nullptr) {

cout << "Очередь пуста ";

}

else {

Element\* temp = start;

start = start->next;

delete temp;

}

}

void printQueue(Element\*& start) {

Element\* now = start;

while (now != nullptr) {

cout << now->data << " ";

now = now->next;

}

cout << endl;

}

void sumNegativeNumber(Element\*& start) {

Element\* now = start;

double sum = 0;

while (now != nullptr) {

if (now->data < 0) {

sum += now->data;

}

now = now->next;

}

cout << "Сумма отрицательных чисел = " << sum;

}

void main() {

setlocale(LC\_ALL, "ru");

Element\* start = nullptr;

Element\* end = nullptr;

enqueue(start, end, -2.1);

enqueue(start, end, 1.3);

enqueue(start, end, -1.34);

enqueue(start, end, 3.3);

printQueue(start);

dequeue(start);

enqueue(start, end, 2.9);

printQueue(start);

sumNegativeNumber(start);

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Задание 3:** Написать программу в соответствии с индивидуальным заданием

Даны две очереди; адреса начала и конца первой равны *P*1 и *P*2, а второй – *P*3 и *P*4 (если очередь является пустой, то соответствующие адреса равны nil). Переместить все элементы первой очереди (в порядке от начала к концу) в конец второй очереди и вывести новые адреса начала и конца второй очереди. Операции выделения и освобождения памяти не использовать.

Код программы:

#include<iostream>

#include<queue>

using namespace std;

struct Element {

int data;

Element\* next;

Element(int value) : data(value), next(nullptr) {}

};

void enqueue(Element\* &start, Element\* &end, int value) {

Element\* newElement = new Element(value);

if (start == nullptr) {

start = end = newElement;

}

else {

end->next = newElement;

end = newElement;

}

}

void mergeQueue(Element\* &P1, Element\*& P2, Element\*& P3, Element\*& P4) {

if (P3 == nullptr) {

P3 = P1;

P4 = P2;

}

else {

Element\* newElemenet = P1;

while (newElemenet != nullptr) {

P4->next = newElemenet;

P4 = newElemenet;

newElemenet = newElemenet->next;

}

}

}

void printQueue(Element\*& start) {

Element\* now = start;

while (now != nullptr) {

cout << now->data << " ";

now = now->next;

}

cout << endl;

}

void printAdress(Element\*& P3, Element\*& P4) {

if (P3 == nullptr) {

cout << "Адрес начала второй очереди " << "nil" << endl;

cout << "Адрес конца второй очереди " << "nil" << endl;

}

else {

cout << "Адрес начала второй очереди " << P3 << endl;

cout << "Адрес конца второй очереди " << P4 << endl;

}}

int main() {

setlocale(LC\_ALL, "ru");

Element\* P1 = nullptr;

Element\* P2 = nullptr;

Element\* P3 = nullptr;

Element\* P4 = nullptr;

int n1;

cout << "Введите длину первой очереди = ";

cin >> n1;

cout << endl;

for (int i = 0; i < n1; i++) {

int now;

cin >> now;

enqueue(P1, P2,now);

}

int n2;

cout << "Введите длину второй очереди = ";

cin >> n2;

cout << endl;

for (int i = 0; i < n2; i++) {

int now;

cin >> now;

enqueue(P3, P4, now);

}

mergeQueue(P1, P2, P3, P4);

printQueue(P3);

printAdress(P3, P4);

}

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Задание 4**: написать программу в соответствии с индивидуальным заданием

1. Создать линейный однонаправленный список из целых чисел. Вставить в список число 10 после каждого элемента с отрицательным значением.

Код программы:

#include <iostream>

#include <list>

void insertTenAfterNegatives(std::list<int>& mylist) {

auto it = mylist.begin();

while (it != mylist.end()) {

if (\*it < 0) {

it = mylist.insert(++it, 10);

}

++it;

}

}

void printList(const std::list<int>& mylist) {

for (const auto& element : mylist) {

std::cout << element << " ";

}

std::cout << std::endl;

}

int main() {

setlocale(LC\_ALL, "ru");

std::list<int> mylist = { 1, -2, 3, -4, 5 };

std::cout << "Исходный список: ";

printList(mylist);

insertTenAfterNegatives(mylist);

std::cout << "Список после вставки числа 10: ";

printList(mylist);

return 0;

}

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

2. Создать линейный однонаправленный список из целых чисел. Вставить в список число 12 после первого элемента большего 10

Код программы:

#include <iostream>

#include <list>

int main() {

setlocale(LC\_ALL, "ru");

std::list<int> myList = { 5, 12, 8, 15, 3 };

std::cout << "Исходный список: ";

for (const auto& element : myList) {

std::cout << element << " ";

}

std::cout << std::endl;

auto it = std::find\_if(myList.begin(), myList.end(), [](int value)

{

return value > 10;

});

if (it != myList.end()) {

myList.insert(++it, 12);

}

std::cout << "Список после вставки числа 12: ";

for (const auto& element : myList) {

std::cout << element << " ";

}

std::cout << std::endl;

return 0;

}

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

**Задание 5**: написать программу в соответствии с индивидуальным заданием

1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент перед каждым элементом со значением -2. Вставить число 33 в конец списка.

Код программы:

#include <iostream>

#include <list>

void removeElementBeforeMinus2(std::list<double>& myList) {

auto it = myList.begin();

while (it != myList.end()) {

if (\*it == -2) {

if (it != myList.begin()) {

it = myList.erase(std::prev(it));

}

}

++it;

}

}

int main() {

setlocale(LC\_ALL, "ru");

std::list<double> myList = { 1.5, -2, 3.7, -2, 4.2, -2, 6.1 };

std::cout << "Исходный список: ";

for (const auto& element : myList) {

std::cout << element << " ";

}

std::cout << std::endl;

removeElementBeforeMinus2(myList);

myList.push\_back(33);

std::cout << "Список после удаления и вставки: ";

for (const auto& element : myList) {

std::cout << element << " ";

}

std::cout << std::endl;

return 0;

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

2. Создать линейный двунаправленный список из целых чисел. Вставить в список число 25 перед каждым элементом с положительным значением. Удалить из списка все отрицательные числа.

Код программы:

#include <iostream>

#include <list>

void insert25BeforePositive(std::list<int>& myList) {

auto it = myList.begin();

while (it != myList.end()) {

if (\*it > 0) {

it = myList.insert(it, 25);

++it;

}

++it;

}

}

void removeNegatives(std::list<int>& myList) {

myList.remove\_if([](int value) { return value < 0;});

}

int main() {

setlocale(LC\_ALL, "ru");

std::list<int> myList = { 3, -2, 8, -5, 10 };

std::cout << "Исходный список: ";

for (const auto& element : myList) {

std::cout << element << " ";

}

std::cout << std::endl;

insert25BeforePositive(myList);

removeNegatives(myList);

std::cout << "Список после вставки и удаления: ";

for (const auto& element : myList) {

std::cout << element << " ";

}

std::cout << std::endl;

return 0;

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Задание 6**: написать программу в соответствии с индивидуальным заданием

Дан указатель P1 на первый элемент непустого двусвязного списка. Удалить из списка все элементы с нечетными значениями и вывести указатель на первый элемент преобразованного списка (если в результате удаления элементов список окажется пустым, то вывести nil). После удаления элементов из списка освобождать память, которую они занимали

Код программы:

#include <iostream>

struct Element {

int data;

Element\* prev;

Element\* next;

Element(int value) : data(value), prev(nullptr), next(nullptr) {}

};

Element\* removeOddNodes(Element\* head) {

Element\* current = head;

Element\* newHead = nullptr;

Element\* tail = nullptr;

while (current != nullptr) {

Element\* nextNode = current->next;

if (current->data % 2 == 0) {

Element\* newNode = new Element(current->data);

if (newHead == nullptr) {

newHead = newNode;

tail = newNode;

}

else {

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

}

delete current;

current = nextNode;

}

return newHead;

}

void printList(Element\* head) {

while (head != nullptr) {

std::cout << head->data << " ";

head = head->next;

}

std::cout << std::endl;

}

void deleteList(Element\*& head) {

while (head != nullptr) {

Element\* temp = head;

head = head->next;

delete temp;

}

}

int main() {

setlocale(LC\_ALL, "ru");

Element\* head = new Element(1);

head->next = new Element(2);

head->next->prev = head;

head->next->next = new Element(3);

head->next->next->prev = head->next;

head->next->next->next = new Element(4);

head->next->next->next->prev = head->next->next;

head->next->next->next->next = new Element(5);

head->next->next->next->next->prev = head->next->next->next;

std::cout << "Исходный список: ";

printList(head);

head = removeOddNodes(head);

if (head != nullptr) {

std::cout << "Преобразованный список: ";

printList(head);

}

else {

std::cout << "nil" << std::endl;

}

deleteList(head);

return 0;

}

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание