Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**Пермский национальный исследовательский политехнический университет**

Факультет Электротехнический Кафедра ИТАС

Специальность Промышленная Робототехника

**ОТЧЁТ**

**о лабораторной работе №3**

Обработка исключительных ситуаций

|  |  |
| --- | --- |
|  | Выполнил:  Студент группы ПРТ-21-1Б  Торган Г.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Проверил:  Доцент кафедры ИТАС Полякова О.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Пермь 2022**

**Постановка задачи:**

Класс- контейнер СПИСОК с ключевыми значениями типа

int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера списка;

\* число – умножает все элементы списка на число;

- n – переход влево к элементу с номером n.

Варианты:

2 , 3

**Текст программы:**

**Lab3.cpp**

#include <iostream>

#include <random>

#include <fstream>

#include <string>

#include "Header.h"

#include "error.h"

using namespace std;

int main()

{

try

{

setlocale(0, "rus");

int n, k, u = 1;

srand(time(0));

list l;

cout << "Сколько элементов создать?" << endl;

cin >> k;

l.pushback(k);

l.print();

cout << "Какой по счету элемент вывести?" << endl;

cin >> n;

l.operator[](n);

cout << "Длинна списка: ";

l.operator()(k);

cout << endl;

cout << "На какое число умножить?" << endl;

cin >> u;

l.operator\*(u);

l.print();

l.alldel();

}

catch (error e)

{

e.what();

}

return 0;

}

**Header.h**

#pragma once

#include <iostream>

#include <random>

#include <fstream>

#include <string>

#include "error.h"

using namespace std;

struct node

{

int data;

node\* next;

node\* prev;

node(int \_data)

{

data = \_data;

next = nullptr;

prev = nullptr;

}

};

class Pair

{

private:

int \_a;

double \_b;

public:

Pair()

: \_a(0), \_b(0) { }

Pair(int a, double b)

: \_a(a), \_b(b) { }

~Pair() { }

////////////////////////////////////////////////////////////////

Pair& operator++ () {

++\_a;

return \*this;

}

Pair& operator++ (int) {

\_b += 1.0;

return \*this;

}

friend std::ostream& operator<< (std::ostream&, const Pair&);

};

std::ostream& operator<< (std::ostream& ost, const Pair& obj) {

return (ost << obj.\_a << " : " << obj.\_b);

}

struct list

{

node\* head;

node\* tail;

list() : head(nullptr), tail(nullptr) {}

bool zero()

{

return head == nullptr;

}

void push\_back(int y)

{

node\* x = new node(y);

if (zero())

{

head = x;

tail = x;

}

tail->next = x;

x->prev = tail;

tail = x;

}

void pushback(int k)

{

for (int i = 0; i < k; i++)

{

node\* x = new node(rand() % 10);

if (zero())

{

head = x;

tail = x;

}

tail->next = x;

x->prev = tail;

tail = x;

}

}

void print()

{

if (zero()) {

cout << "список пуст";

return;

}

node\* current\_node = head;

while (current\_node)

{

cout << current\_node->data << " ";

current\_node = current\_node->next;

}

cout << endl;

}

node\* operator[] (int index)

{

--index;

if (index == 0)

{

/\*node\* p = head;

p = p->next;

p->prev = nullptr;

free(head);

head->next = p->next;

head = p;\*/

print();

}

else {

if (zero()) return nullptr;

node\* p = head;

node\* x = head;

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

p = p->next;

if (!p) return nullptr;

}

for (int i = 0; i < (index - 1); i++) {

x = x->next;

if (!x) return nullptr;

}

/\*x->next = p->next;

delete p;

\*/

node\* current\_node = p;

cout << current\_node->data << " ";

return 0;

}

}

node\* operator() (int lng)

{

cout << lng << endl;

return 0;

}

node\* operator\* (int um)

{

if (zero()) {

cout << "список пуст";

return 0;

}

node\* current\_node = head;

while (current\_node)

{

current\_node->data = (current\_node->data) \* um;

current\_node = current\_node->next;

}

return 0;

}

node\* operator- (int n)

{

if (zero()) {

return 0;

}

node\* p = tail;

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

p = p->prev;

if (!p) return nullptr;

}

return 0;

}

void alldel()

{

node\* current;

current = head;

for (int i = 1; head->next = nullptr; i++) {}

if (zero()) return;

node\* current\_node = head;

head = current\_node->next;

delete current\_node;

}

};

**error.h**

#pragma once

#include <string>

#include <iostream>

using namespace std;

class error //класс ошибка

{

string str;

public:

//конструктор, инициирует атрибут str сообщением об ошибке

error(string s) { str = s; }

void what() { cout << str << endl; } //выводит значение атрибута str

};

**Ответ для варианта №14**

