Пермский национальный исследовательский политехнический университет.

Лабораторная работа № 9 по ООП.

«Обработка исключительных ситуаций».

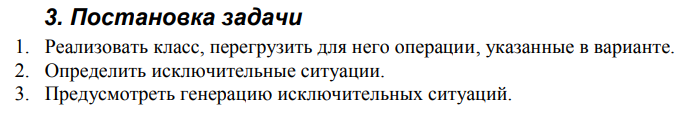
Выполнил: студент группы РИС-23-2б

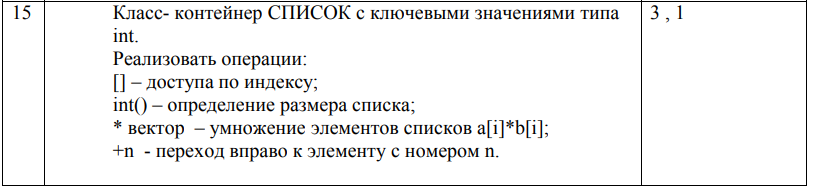
Вековшинин Иван Николаевич

Проверила: доцент кафедры ИТАС

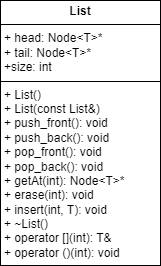
О.А. Полякова.

2024 г.





UMLDiagram:



Код:

**List.h**

#pragma once

#include<iostream>

#include<vector>

using namespace std;

template <typename T>

struct Node

{

T data;

Node\* next\_ptr;

void addData(T Data)

{

data = Data;

next\_ptr = nullptr;

}

};

template <typename T>

struct List

{

Node<T>\* head;

Node<T>\* tail;

int size;

List() {

this->head = nullptr;

this->tail = nullptr;

this->size = 0;

}

List(const List& a) {

this->size = a.size;

Node<T>\* temp\_ptr = head;

for (Node<T>\* i = a.head; i != nullptr; i = i->next\_ptr) {

temp\_ptr->data = i->data;

temp\_ptr = temp\_ptr->next\_ptr;

}

}

void pop\_front()

{

if (head == nullptr) return;

if (head == tail)

{

delete tail;

head = tail = nullptr;

return;

}

Node<T>\* node = head;

head = node->next\_ptr;

delete node;

--size;

}

void push\_back(T data)

{

Node<T>\* node = new Node<T>;

node->addData(data);

if (head == nullptr) head = node;

if (tail != nullptr) tail->next\_ptr = node;

tail = node;

++size;

}

void push\_front(T data)

{

Node<T>\* node = new Node<T>;

node->addData(data);

node->next\_ptr = head;

head = node;

if (tail == nullptr) tail = node;

++size;

}

void pop\_back()

{

if (tail == nullptr) return;

if (head == tail)

{

delete tail;

head = tail = nullptr;

return;

}

Node<T>\* node = head;

for (; node->next\_ptr != tail; node = node->next\_ptr);

node->next\_ptr = nullptr;

delete tail;

tail = node;

--size;

}

Node<T>\* getAt(int k) {

if (k < 0) return 0;

Node<T>\* node = head;

int n = 0;

while (node && n != k && node->next\_ptr) {

node = node->next\_ptr;

n++;

}

if (n == k) return node;

else return nullptr;

}

void insert(int k, T data) {

Node\* left = getAt(k);

if (left == nullptr) return;

Node\* right = left->next\_ptr;

Node<T>\* node = new Node<T>;

node->addData(data);

left->next\_ptr = node;

node->next\_ptr = right;

if (right == nullptr) tail = node;

size++;

}

void erase(int k) {

if (k < 0) return;

if (k == 0) {

pop\_front();

return;

}

Node\* left = getAt(k - 1);

Node\* node = left->next\_ptr;

if (node == nullptr) return;

Node\* right = node->next\_ptr;

left->next\_ptr = right;

if (node == tail) tail = left;

delete node;

--size;

}

~List()

{

while (head != nullptr) pop\_back();

}

T& operator [] (int a) {

if (a >= size || a < 0) throw "Выход за границы списка!";

return getAt(a)->data;

}

void operator () (int a) {

if (a <= 0) throw "Ошибка создания списка";

this->size = a;

Node<T>\* node = new Node<T>;

head = node;

Node<T>\* temp\_ptr = head;

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

Node<T>\* node = new Node<T>;

temp\_ptr->next\_ptr = node;

temp\_ptr = temp\_ptr->next\_ptr;

}

tail = temp\_ptr;

temp\_ptr->next\_ptr = nullptr;

}

class iterator {

Node<T>\* it;

int Size;

public:

iterator(List& a) {

this->it = a.head;

this->Size = a.size;

}

Node<T>\* operator + (int b) {

if (b >= Size || b < 0) throw "Выход за границы списка";

Node<T>\* iter = it;

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

iter = iter->next\_ptr;

return iter;

}

};

friend List& operator\*(List& a, vector<int> b);

};

template <typename T, typename T2>

List<T>& operator\*(List<T>& a, vector<T2>& b) {

if (a.size == b.size()) {

for (int i = 0; i < a.size; ++i) {

a.getAt(i)->data = a.getAt(i)->data \* b[i];

}

}

return a;

}

Main.cpp

#include<iostream>

#include"List.h"

using namespace std;

int main() {

system("chcp 1251 > null");

List<int> list;

try

{

list(5);

}

catch (const char\* a)

{

cout << a;

}

List<int>::iterator a(list);

list[4] = 5;

try

{

cout << (a + 4)->data;

}

catch (const char\* a)

{

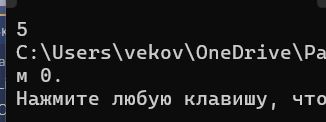
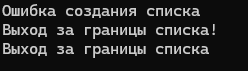
cout << a;

}

return 0;

}

Результат работы программы:



Ответы на вопросы:

1)Исключение – это непредвиденное или аварийное событие.

2)Исключения позволяют разделить вычислительный процесс на 2 части: обнаружение аварийное ситуации, обработка аварийной ситуации. Достоинтсва такого подхода: удобно использовать в программе, не требуется возвращать значение в вызываемую функцию.

3)Исключение генерируется оператором throw.

4)Контролируемый блок – try. В него программист пишет код, в котором возможна исключительная ситуация.

5)Секция-ловушка – catch. Нужна для обработки исключения, если возникла аварийная ситуация.

6)Форма записи секции-ловушки: catch (спецификация\_исключения), где спецификация может иметь три формы: (тип имя), (тип), (...). Первая и вторая форма соответственно обрабатывают конкретные исключения, а третья форма перехватывает все исключения, которые еще не были обработаны.

7)Класс – exception.

8)Можно определять собственные исключения, унаследовав их от класса exception. Для создания собственной иерархии исключений надо объявить свой базовый класс-исключения, и унаследовать его от класса exception.

9)Те исключения, которые в блоке catch описаны со спецификацией int или double.

10)Все исключения, которые не обработаны программистом.

11) Исключение может генерироваться в той части программы, которая обнаружила аномальную ситуацию.

12)без спецификаци исключений

|  |
| --- |
| #include<iostream>  #include<math.h>  using namespace std;  double square(double a, double b, double c) {  double p = (a + b + c) \* 0.5;  double square = p \* (p - a) \* (p - b) \* (p - c);  return sqrt(square);  }  int main() {  system("chcp 1251 > null");  cout << square(5, 6, 7);  } |

Со спецификацией throw:

|  |
| --- |
| #include<iostream>  #include<math.h>  using namespace std;  double square(double a, double b, double c) {  double p = (a + b + c) \* 0.5;  if (p < 0) throw 0;  double square = p \* (p - a) \* (p - b) \* (p - c);  return sqrt(square);  }  int main() {  system("chcp 1251 > null");  try  {  square(-5, -6, -7);  cout << square(-5, -6, -7);  }  catch (...)  {  cout << "Данные о треугольнике введены неправильно!";  }  } |

С конкретной спецификацией:

|  |
| --- |
| #include<iostream>  #include<math.h>  using namespace std;  double square(double a, double b, double c) {  double p = (a + b + c) \* 0.5;  if (p < 0) throw "Данные о треугольнике введены неправильно!";  double square = p \* (p - a) \* (p - b) \* (p - c);  return sqrt(square);  }  int main() {  system("chcp 1251 > null");  try  {  square(-5, -6, -7);  cout << square(-5, -6, -7);  }  catch (const char\* a)  {  cout << a;  }  } |

Спецификация с собственным релизованым исключением:

|  |
| --- |
| #include<iostream>  #include<math.h>  #include<string>  using namespace std;  class SomeException : public std::exception{  string error;  public:  SomeException(string error) {  this->error = error;  }  const char\* what() {  return error.c\_str();  }  };  double square(double a, double b, double c) {  double p = (a + b + c) \* 0.5;  if (p < 0) throw SomeException("Данные о треугольнике введены неправильно!");  double square = p \* (p - a) \* (p - b) \* (p - c);  return sqrt(square);  }  int main() {  system("chcp 1251 > null");  try  {  square(-5, -6, -7);  cout << square(-5, -6, -7);  }  catch (SomeException& ex)  {  cout << ex.what();  }  } |