Учебный курс: Технологии программирования

Кафедра: ИС Факультет: ИТиП

Лабораторная работа №8-9. Шаблоны, исключения.

Выполнил: Трофимов В.А. Группа: 1511

Преподаватель: Повышев В.В.

# Задание

Написать шаблонную функцию или класс согласно варианту. Описать класс-исключение или иерархию классов-исключений. Генерировать исключения в соответствующих исключительных ситуациях. Если у вас есть другие предложения по исключительным ситуациям – используйте их.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D | | Вычисляет N-ю (целую) степень числа | | | Степень (показатель) – целое число. Тип переменной. |
| G | | Очередь из макс. N элементов типа T | | int N, class T | Переполнение очереди. Изъятие элемента из пустой очереди. | | |

# Входные/Выходные данные

## Консоль

2 ^ 3 = 8

0.5 ^ 2 = 0.25

You tried to add something to a full queue.

41 18467 6334 26500 19169 15724 11478 29358 26962 24464 5705 28145 23281 16827 9961 491 2995 11942 4827 5436 32391 14604 3902 153 292 12382 17421 18716 19718 19895 5447 21726 14771 11538 1869 19912 25667 26299 17035 9894 28703 23811 31322 30333 17673 4664 15141 7711 28253 6868 You tried to get something out from an empty queue. return 0. 0

# Исходный текст

## Файл main.cpp

#include <cmath>

#include <iostream>

#include <string>

#include <exception>

using namespace std;

template<typename baseType, int exp>

baseType power(const baseType baseType) {

return pow(baseType, exp);

}

class popFromEmptyQueueException : public exception {

public:

popFromEmptyQueueException() {

}

int what() {

cout << "You tried to get something out from an empty queue. return 0.";

return 0;

}

};

class pushToFullQueueException : public exception {

public:

pushToFullQueueException() {

}

int what() {

cout << "You tried to add something to a full queue.";

return 0;

}

};

template<int N, class T> class queue {

private:

struct data {

T value;

data \* next;

};

data \* last;

data \* first;

int size;

public:

queue() {

last = NULL;

first = NULL;

size = 0;

}

bool isEmpty() {

if (size == 0) return true;

else return false;

}

bool isFull() {

if (size == N) return true;

else return false;

}

T Pop() {

if (size == 0) {

throw popFromEmptyQueueException();

} else {

T temp = first -> value;

data \* next = first -> next;

delete(first); size--;

first = next;

return temp;

}

}

void Push(const T value) {

if (size == 0) {

data \* temp = new data;

first = temp;

last = temp;

first -> value = value;

first -> next = NULL;

size = 1;

} else {

if (size == N) throw pushToFullQueueException();

data \* temp = new data;

temp -> value = value;

temp -> next = NULL;

last -> next = temp;

last = temp;

size++;

}

}

};

int main() {

cout << "2 ^ 3 = " << power<int, 3>(2) << endl;

cout << "0.5 ^ 2 = " << power<double, 2>(0.5) << endl;

queue<50, int> q;

for (int i = 0; i < 51; i++) {

try {

q.Push(rand());

} catch (pushToFullQueueException & e) {

cout << e.what();

}

}

for (int i = 0; i < 51; i++) {

try {

cout << q.Pop() << " ";

} catch (popFromEmptyQueueException & e) {

cout << e.what();

}

}

return 0;

}