Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЁТ**

**по дисциплине «Информатика»**

Семестр: 2

На тему: «Обработка исключительных ситуаций»

Выполнил студент ИВТ-22-2б:

Зайцев Павел Денисович

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

1. Реализовать класс, перегрузить для него операции, указанные в варианте.
2. Определить исключительные ситуации.
3. Предусмотреть генерацию исключительных ситуаций.

**Варивант 9:**

Класс- контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

* []-достапа по индексу;
* !=-проверки на неравенство;
* < число- принадлежность числа множеству;
* +n -переход вправо к элементу с номером n.

Варианты реализации 1,3.

**Словесное описание исключительных ситуаций**

Исключительные ситуации генерируются:

1, 2 – в операции [] – при попытке обратиться к элементу с номером меньше 0 или больше текущего размера множества;

3 – +n– в операции – при попытке перейти к элементу, который находится вне границ множества.

**Определение класса *error* или иерархии пользовательских классов для определения исключительных ситуаций**

Класс Error

Методы типа public:виртуальная функция what().

Класс IndexError

Поля типа private: msg-строчная переменная.

Методы типа public:Функция IndexError(), и виртуальная функция what().

Класс SizeError

Поля типа private: msg-строчная переменная.

Методы типа public:Функция SizeError(), и виртуальная функция what().

Класс MaxSizeError

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция MaxSizeError(), и виртуальная функция what().

Класс EmptySizeError

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция EmptySizeError(), и виртуальная функция what().

Класс IndexError1

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция IndexError1(), и виртуальная функция what().

Класс IndexError2

Поля типа private: msg\_-строчная переменная.

Методы типа public:Функция IndexError2(), и виртуальная функция what().

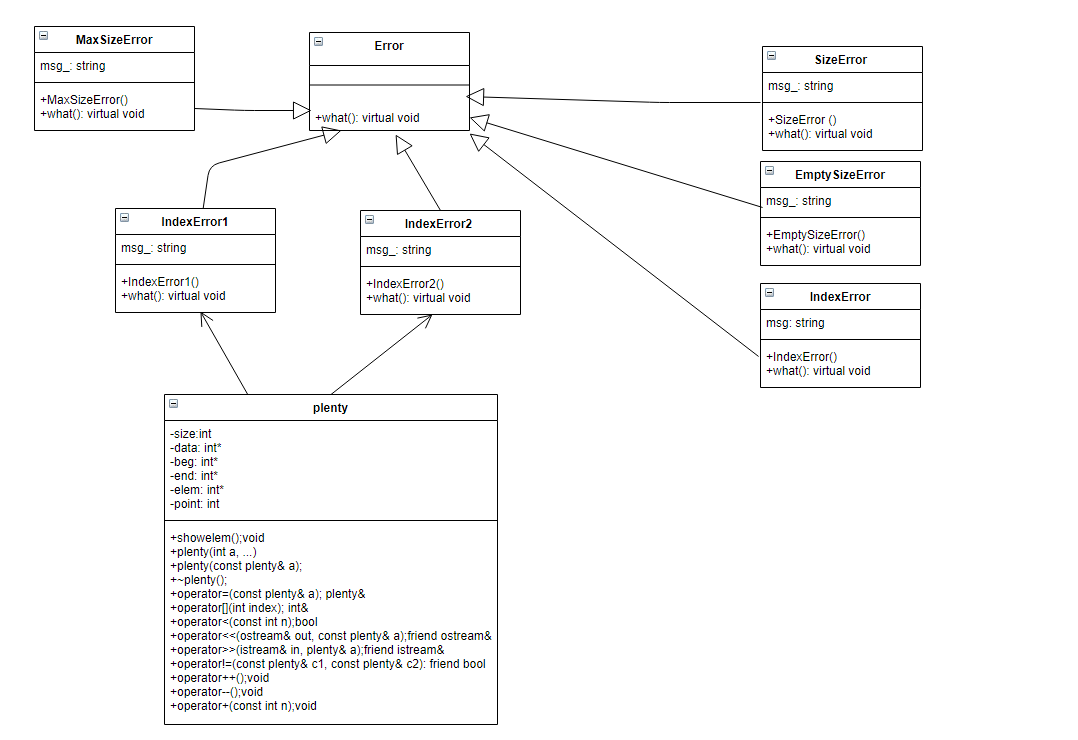
**Класс контейнер**

Класс plenty

Поля типа private:Целочисленные указатели data,beg,end,elem. Целочисленные переменные size,point.

Методы типа public:2 конструктора, деструктор, оператор присваивания operator=(const plenty& a),операторы потокового ввода вывода operator<<(ostream& out, const plenty& a), operator>>(istream& in, plenty& a), операция инкремента и декремента operator--(), operator++(), .

Также описание классов представлено на UML-диаграмме.



**Определение компонентных функций для класса-конетейнера.**

Функция доступа по индексу:

int& plenty::operator[] (int index)

{

if (index < 0) throw IndexError1() ;//если индекс отрицательный, то генерируется исключение

//если индекс больше размер вектора, то генерируется исключение

if (index >= size) throw IndexError2();

return data[index];

}

Функция проверки на неравество:

bool operator!=(const plenty& c1, const plenty& c2) {

if (c1.size != c2.size) {//Сравнение длинн

return 1;

}

else {

for (int i = 0; i < c2.size; i++) {//Сравнение элементов

if (c1.data[i] != c2.data[i]) {

return 1;

}

}

return 0;

}

}

Функция принадлежности числа множеству:

bool plenty::operator<(const int n) {

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

if (data[i] == n) {

return 1;

}

}

return 0;

}

Функция перевода указателя на n элементов:

void plenty:: operator+ (const int n) {

if (point + n > size - 1) throw 1;

else {

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

++elem;

}

}

}

**Функция main()**

**Вариант 1**

int main() {

//контролируемый блок

try

{

plenty x(2,10);//множество из двух элементов

plenty y(15,10);//пустой множества

x[0] = 13;

cout << y[-3];//Выход за границу множества

++x;

++x;//Выход за границу множества

y + 20;//Выход за границу множества

x.showelem();

}

//обработчик исключения

catch (int)

{

cout << "ERROR!!!" << endl;

}//сообщение об ошибке

return 0;

}

**Вариант 3**

int main() {

try

{

plenty x(4,10,25,5416,25);

plenty y(6,25,35,15,12,11,11);

cout << x;

cout <<"\nx=" << x[1] << endl;

cout <<endl<< y;

--x;

--x;//Выход за границы множества

cout << x;

y + 20;//Выход за границы множества

}

catch (Error& e)

{

e.what();

}

return 0;

}

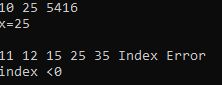
**Результаты работы программ**

**Вариант1**



Выведется ошибка, потому что одна из функций создаст и отправит исключение.

**Вариант2**



Выведется ошибка доступа по индексу, потому что одна из функций создаст и отправит исключение класс обработчик событий.

**Контрольные вопросы**

1. *Что представляет собой исключение в С++?*

Исключение – это непредвиденное или аварийное событие.В С++ исключение – это объект, который система должна генерировать привозникновении исключительной ситуации. Генерация такого объекта и создаетисключительную ситуацию.

1. *На какие части исключения позволяют разделить вычислительный процесс?*

*Достоинства такого подхода?*

1. *Обнаружение аварийной ситуации.*
2. *Обработка аварийной ситуации.*
3. *Какой оператор используется для генерации исключительной ситуации?*

throw.

1. *Что представляет собой контролируемый блок? Для чего он нужен?*

try{

//Блок программы

}

Именно в нем должны создаваться исключения, поэтому генератор исключений throw работает только в нем.

1. *Что представляет собой секция-ловушка? Для чего она нужна?*

сatch () *{*

//Блок программы

*}*

Нужен для обработки исключений.

1. *Какие формы может иметь спецификация исключения в секции ловушке? В каких ситуациях используются эти формы?*

Сatch( спецификация исключения ), где спецификация исключения может иметь

три формы:

1) (тип имя)

2) (тип)

3) (…)

Формы 1 и 2 обрабатывают конкретные исключения, а форма 3 перехватывает все

исключения, такую ловушку надо помещать последней, тогда она будет обрабатывать все исключения, которые еще не были обработаны.

1. *Какой стандартный класс можно использовать для создания собственной иерархии исключений?*

Class exception

1. *Каким образом можно создать собственную иерархию исключений?*

Для создания собственной иерархии исключений надо объявить свой базовый классисключение, например:class Error{};

Остальные классы будут наследниками этого класса, аналогично тому, как это сделано виерархии стандартных исключений:class Child\_Error:public Error{}; class Parents\_Error:public Error{};

1. *Если спецификация исключений имеет вид: void f1()throw(int,double); то какие*

*исключения может порождать функция f1()?*

Только исключения типа int и double.

1. *Если спецификация исключений имеет вид: void f1()throw(); то какие исключенияможет порождать функция f1()?*

Исключения абсолютно любого типа.

1. *В какой части программы может генерироваться исключение?*

В операторе throw.

1. Написать функцию, которая вычисляет площадь треугольника по трем сторонам (формула Герона).

* Без спицификации исключений

int geron(int a, int b, int c) {

int S, P;

P = (a + b + c) / 2;

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}

* Со спицификатором throw

double geron(int a, int b, int c) {

if (a \* b \* c < 0) throw 1;

double P;

double S;

P = ((a + b + c) / 2);

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}

* C собственным реализованным исключением.

void F() {

exit(1);

}

void set\_terminate() {

F();

}

double geron(int a, int b, int c) {

if (a \* b \* c < 0) set\_terminate();

double P;

double S;

P = ((a + b + c) / 2);

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}

* С конкретной спецификацией с подходящим стандартным исключением.

double geron(int a, int b, int c) {

if (a \* b \* c < 0) terminate();

double P;

double S;

P = ((a + b + c) / 2);

S = sqrt(P\*(P-a)\* P \* (P - b)\* P \* (P - c));

return S;

}