**Лабораторная работа № 7**

**Тема**: Исключительные ситуации. Замена фукции unexpected () и terminate ().

**Цель**: Научиться использовать технику обработки исключительных ситуаций.

**Задание**

1. Наберите текст программы приведенного примера обработки исключительных ситуаций и наладьте его (рассмотрите 2 случая работы программы). Объясните полученные результаты в первом случае и во втором.
2. Напишите программу, в которой пользователю предлагают несколько раз ввести число. Каждый раз выводить его корень. Если вводится отрицательное число, выводить сообщение об ошибке. Использовать технику исключительных ситуаций.
3. Наберите текст программы приведенного примера обработки исключительных ситуаций и наладьте его (рассмотрите 2 случая работы программы). Объясните полученные результаты в первом случае и во втором.

/\*\*Лабараторная работа №6

\*\*\*Задание № 1

\*\*\*Создать структуру родового связанного списка, используя шаблоны.

\*\*\*КНТУ МТФ КИ-15 Аннаев Арслан

\*\*\* Листинг OOP\_Lab\_\_1

\*/

#include <iostream>

#include <cstdlib>

#include <exception>

#define MAXERR 10

using namespace std;

class MaxErr{};

// класс exception

class Error{

public:

Error();

void Say();

private:

static int count;

};

void Run() throw(Error); // прототип функции-обработчика:

void trapper(); // прототип функции-обработчика:

void zapper(); // прототип функции-обработчика:

int Error::count; // количетсво ошибок

int main()

{

set\_unexpected(trapper); // установка обработчика исключ.

set\_terminate(zapper); // установка обработчика завершения программы

for(;;)

{

try {

Run(); // функция в блоке try

}

catch(Error e) // перехват исключения

{

e.Say(); // обработка исключения методом класса Error

}

return 0;

}

}

void Run()throw(Error) // функция в которой обрабатываются только Error

{

throw Error();

//throw "Unknown object type"; //

}

void trapper() //функции-обработчика

{

cout << "Handler contingencies. .";

throw Error();

}

void zapper() // прототип функции-обработчика:

{

cout << "Completion handler function";

exit(-1);

}

Error::Error() // конструктор класса

{

count++;

if(count>MAXERR)

throw MaxErr();

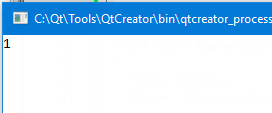
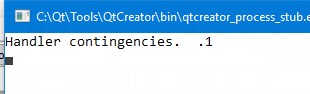
}

void Error::Say() // метод класса Error

{

cout << count << endl; // вывод количества ошибок

}



1. Напишите программу, в которой пользователю предлагают несколько раз ввести число. Каждый раз выводить его корень. Если вводится отрицательное число, выводить сообщение об ошибке. Использовать технику исключительных ситуаций.

/\*\*Лабараторная работа №6

\*\*\*Задание № 2

\*\*\*Создать структуру родового связанного списка, используя шаблоны.

\*\*\*КНТУ МТФ КИ-15 Аннаев Арслан

\*\*\* Листинг OOP\_Lab\_\_2

\*/

#include <iostream> // библиотека ввода вывода

#include <cmath> // матем. библиотека

*using* *namespace* std; // пространство имен

int main()

{

double first; // переменная для хранения вводимого значения

double result; // переменная для хранения результата

char ch; // переменная для хранения условия выхода

*do* {

cout << "Please Enter the value \n" << endl; // приглашение

cin >> first; // ввод числа

*try*{ //блок исключ. ситуаций

*if* (first < 0) // если отрицательное число

*throw* "Error" ; // возбуждение искл. ситуац

result = sqrt(first); // корень числа

cout << "Result " << result << endl; // вывод результата

}

*catch*(*const* char\* message) // обработчик искл. ситуаций

{

cout << "Negative value = " << message; // отображение сообщ.

}

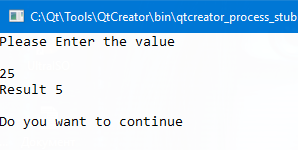
cout << "\nDo you want to continue \n" << endl; // повтор при условии “Y”

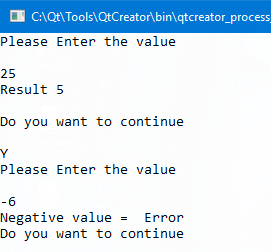
cin >> ch;

} *while*(ch == 'Y');

*return* 0;

}





Контрольные вопросы к Л.Р. 7:

1. Что представляет собой исключительная ситуация?

*Это условие исключительной ситуации, что требует специальной обработки.*

1. Что называют обработчиками исключений?

*Операторы, перехватывающие исключительные ситуации, называются обработчиками исключений*

1. Что представляет собой блок try?

*Блок try содержит один или несколько операторов, исключительные ситуации которые нужно перехватывать*

1. На что указывает следующий за именем функции и списком параметров выражение throw ()?

*Возбуждение искл. Ситуации и передачей параметра в блок catch()*

1. Какие и. с. называют непредвиденными?

*Исключительные ситуации, которые не обрабатываются одним оператором catch, называются непредвиденными исключительными ситуациями.*

1. Что происходит с необработанными исключительными ситуациями?

*Вызывается одна из трех специальных функций, которые автоматически присоединяются к каждой программе на С ++, использующий исключительные ситуации. Эти функции имеют имена unexpected () terminate (), abort ().*

1. Как вызывается функция abort (), что она выполняет, можно ли ее заменить своим кодом?

*Функцией terminate(); нет ее нельзя заменить*

1. Для чего используют функции set\_unexpected () set\_terminate (), сколько они имеют аргументов и что возвращают?

*Для задания адреса собственной функции-обработчика непредвиденных исключений, не имеют параметров и ничего не возвращать*

**Теоретические** **сведения**.

Исключительная ситуация - это условие исключительной ситуации, что требует специальной обработки. В.с. лучше всего использовать для обработки ошибок, возникающих при исполнении, но их применение на этом не ограничивается.

Для возбуждения в.с. оператор посылает объект, описывающий суть в.с. Объект м.б. литерально значением, строкой, объектом класса или любого другого объекта. Объект в.с. не обязательно быть объектом класса.

Для обработки в.с. некоторый оператор захватывает условие, посланная другим процессом. Операторы, перехватывают в.с., называются обработчиками в.с.

Программы готовятся к перехвату в.с., испытывая один или несколько процессов, возбуждающие в.с. Для использования в.с. испытывается один или несколько операторов и перехватывается любая в.с., что возбуждается этими операторами.

Обнаружив состояние ошибки, функция может возбудить в.с., что вызывает такие последствия:

Функция объявляет, что возникла условие в.с. Это м.б. как ошибкой, так и другим обстоятельством, что требует обработки.

Функция спрашивает решения проблемы обработки в.с. Обработчик, если он существует, вызывается автоматически в ответ на посылку объекта в.с.

В программе возбуждается в.с. посредством выполнения оператора throw, обычно внутри функции:

throw "Overflow";

В другом месте программы обработчик строчных в.с. может поймать и запечатлеть на экране послан объект. В обработчике задается тип объекта в выражении:

catch (const char \* message)

{

cout << "Error! "<< Message;

....

}

Оператор catch ловит посланы объекты строчных в.с. и отражает их оператором вывода в поток. Если дальнейшие действия не определены в блоке catch, то программа продолжит свою работу после catch. Можно также аварийно прервать выполнение программы, вызвать другую функцию или продолжить цикл для повторного выполнения действий, вызвавших проблему.

В.с. - Механизм для сообщений и принятие мер в случае возникновения условия в.с. В.с. не навязывают необходимых действий. Их обработка полностью зависит от программиста.

Функции в.с. не ограничены обработкой ошибок. Например, пустой учетный объект может сообщать о том, что он пуст, путем возбуждения в.с. Произошла ошибка или что-то другое, зависит только от того, как программист назначит аварийное прерывание выполнения программ.

Но поскольку использование в.с. может привести к двусмысленности в программах, их лучше использовать для принятия мер в случае возникновения реальных ошибок, которые могут заставить программу прервать процесс и привести к аварийному завершению или неверных результатов.

Функции могут возбуждать одну или несколько в.с. различных типов, представляющие различные условия в.с.

Возбуждение в.с. немедленно завершает выполнение функции, в которой выполняется оператор throw.

Исключительные ситуации обеспечивают альтернативный механизм возврата для функций.

#### Объявления в.с.

С помощью альтернативной формы объявления функции можно задавать типы в.с., которые разрешается возбуждать в данной функции. Например, функция AnyFunction () может задать типы своих в.с. таким образом:

void AnyFunction () throw (Error)

Следующий за именем функции и списком параметров выражение throw () указывает, что функция AnyFunction () может возбуждать в.с. типа Error. Это объявление указывает компилятору, что функции AnyFunction запрещается возбуждать в.с. других типов. Для задания функции, возбуждает в.с. нескольких типов, стоит перечислить типы локальных объектов В.С .:

void AnyFunction () throw (Error, char \*, OtherType)

#### Необработанные в.с.

Необработанные в.с. передаются вверх по цепочке вызовов одних функций другими, пока не встретится соответствующий им оператор catch или пока больше не останется неоглянутих обработчиков в.с. Если произойдет последнее, для обработки в.с. вызывается одна из трех специальных функций, автоматически присоединяются к каждой программе на С ++, использующий в.с. Эти функции имеют имена

unexpected () terminate (), abort ().

Они вызываются в соответствии с правилами:

Если в программе возникли в.с., обрабатываемых оператором catch, то вызывается функция unexpected (). В.с., которые не обрабатываются одним оператором catch, называются непредвиденными в.с. По умолчанию unexpected () вызывает функцию terminate (). Непредвиденные в.с. могут возбуждаться программой при обнаружении повреждения стека дескриптором класса, в результате чего вызывается функция terminate (). По умолчанию функция terminate () вызывает функцию abort ().

Функция abort () немедленно прерывает выполнение программы. Она никогда не вызывается непосредственно. Если не обработаны все возможные в.с. и не приняты меры для перепрограммирования функций unexpected () и terminate (), необработанная в.с. аварийно прервет выполнение программы путем обращения к abort ().

Можно заменить функции unexpected () и terminate () своим кодом для обработки необработанных в.с. Например, имеет смысл заменить функцию unexpected () для сообщения о любых необработанные в.с. на этапе разработки программы. Это может помочь выявить недостающие обработчики ошибок в программе. В других случаях, можно заменить terminate () Диагностируя кодом для просмотра памяти, чтобы выявить операторы, которые разрушают кучу.

Заменить функцию abort () нельзя. Ее вызов всегда приводит к завершению программы.

#### Замена функций unexpected () и terminate ()

Для задания адреса собственной функции - обработчика непредвиденных в.с. следует использовать функцию set\_unexpexted (), объявленной в заголовочный файл EXCEPT.H. Функция должна иметь тип unexpected\_function, не иметь аргументов и ничего не возвращать.

Обработчик пользователя непредвиденных в.с. может возбуждать исключительную ситуацию. В этом случае поиски оператора catch начинается с того места, откуда сначала было вызвано обработчик.

Для задания собственной функции-обработчика завершения программы следует использовать функцию set\_terminate () также объявленную в заголовочный файл EXCEPT.H. Функция должна иметь тип terminate\_function, не иметь параметров, ничего не возвращать. Обе функции set\_expected () и set\_termilate () возвращают адрес текущей функции-обработчика. Можно сохранить и затем восстановить существующие обработчики, запомнив их адреса в переменных, а затем передавая их обратно функциям. Например, сначала объявляется прототип функции-обработчика: void unexpectedHandler (); Затем устанавливается обработчик:

set\_unexpected (unexpectedHandler)

Хранение и восстановление адреса старой функции-обработчика:

unexpected\_function savedAddress;

savedAddress = set\_unexpecred (unexpectedHandler) // Новый обработчик

set\_unexpected (saveAddress) // восстановления

// Новый обработчик больше не используется

Функция пользователя terminate () устанавливается аналогично, только вместо set\_unexpected () вызывается функция set\_terminate (); что возвращает адрес типа terminate\_function.

Пример программы установки обработчиков непредвиденных в.с. и завершение. В программе также демонстрируется приемлемый способ обработки непредвиденных в.с. неизвестных типов, которые могут возбуждаться в плохо документированных библиотечных функциях.

#include <iostream.h>

#include <except.h>

#define MaxErr 10

класс MaxErr {};

Класс ошибки {

общественности:

  Ошибка();

  аннулированию Say ();

частный:

  статического счетчика INT;

};

аннулированию Run () бросок (ошибка);

недействительным траппер ();

недействительным заппер ();

INT Error :: кол;

пустая функция()

{

  set\_unexpected (ловец);

  set\_terminate (Zapper);

  для(;;)

  {

    попробуйте {Run ();}

    поймать (Ошибка е) {e.Say ();

  }

}

аннулируются Run () бросок (ошибка)

{

// Ошибка броском ();

  бросить "Невідомий тип об'єкта";

}

недействительным траппер ()

{

  соиЬ << "Обробник непередбачених ситуаций. ".;

  бросить Error ();

}

заппер недействительными ()

{

  соиЬ << "Обробник завершення функції";

  выход (-1);

}

Ошибка :: Error ()

{

  рассчитывать ++;

  если (счетчик> MaxErr)

  бросить MaxError ();

}

аннулируются Error :: Say ()

{

  << соиЬ кол << "\ п";

}

При запуску unexpected.cpp виведеться 10 повідомлень про помилки перед завершенням.

У програмі використовуються два класи в.с. Об'єкт класу MaxError, що не має даних і функцій, посилається, коли число помилок перевищує константу MaxErr, задану рівною 10.

Класу Ошибка Конструктор інкрементує статичний член класу кол. Оскiльки кол член - статичний, то існує тільки один екземпляр цього значення і він існує доти, доки програма не завершиться. ЧЛЕН кол - закритий член класу, отже він не зміниться в операторах програми поза класом. Функція-член Say () відображає значення лічильника помилок програми.

Якщо член число> = MaxErr, конструктор класу Ошибка збуджує в.с. MaxError типу. При збудженні в.с. об'єкт класу не створюється, тому створення об'єкта класу переривається Ошибка.

У функції главный () встановлюються функції-обробники, що заміщають за замовченням функції неожиданное () і прекращается (). Потім виконується нескінчений цикл для. Всередині циклу в блоці попробовать викликається функція Run () і оператор поймать перехоплює усі об'єкти класу Ошибка, що посилає Run (). При збудженні в.с. поймать відображає поточне значення лічильника помилок шляхом звертання до функції Say () посланого об'єкту.

Функція Run () завжди збуджує в.с., моделюючи виникнення декількох помилок. Користувацький обробник непередбачених в.с. траппер () виводить повідомлення про виклик цієї функції. Це відбувається після збудження 10 в.с. Обробник непередбачених в.с. може збудити ще одну в.с. для продовження програми. У прикладі оператор броска Error (); посилає новий екземпляр класу Ошибка. Створення об'єкта класу Ошибка змушує конструктор цього класу збудити ще одну ситуацію типу MaxError. Цей тип помилки не підтримується в операторі поймать, тому викликається обробник завершення програми заппер (). Функція завершення не може збудити в.с., а також не може виконати оператор возврата.

Перепрограмуємо функцію Run () так, щоб у ній збуджувалася в.с. невідомого типу. Наприклад, у якості невідомого типу може виступити рядок.

Коли програма зкомпілює і запустить обробник непередбачених ситуацій, ловец (), буде викликатись для кожної виключної ситуації, збуджуваною функцією Run (). Це відбувається тому, що не існує оператора улов для об'єктів в.с. символ \* типу постановочные Const символ \*. Функція траппер (), проте, транслює нерозпізнані об'єкти в.с., посилаючи об'єкт відомого типу, у даному випадку, ошибка. Нова збуджена в.с. продовжує виконання програми в операторі поймать всередині функції главный (), що обробляє трансльовану виключну ситуацію.

Зрештою, у програмі максимальне число помилок буде перевищено, і об'єкт класу MaxError буде посланий конструктором Ошибка. У результаті викличеться обробник завершення програми шляхом звертання до бібліотечної функції выхода ().

Лістинг результату в 1м випадку (у Run () використовується бросить Error ();)

1

2

...

10

Обробник непередбаченої помилки.

Обробник завершення функції.

У 2му випадку (у Run () - рядок):

Обробник невідомої помилки.

1

Обробник невідомої помилки.

2

...

10

Обробник невідомої помилки.

Обробник завершення функції.

Контрольные вопросы к Л.Р. 7:

1. Что представляет собой исключительная ситуация?

2. Что называют обработчиками исключений?

3. Что представляет собой блок try?

4. На что указывает следующий за именем функции и списком параметров выражение throw ()?

5. Какие у. с. называют непредсказуемыми?

6. Что происходит с необработанными исключительными ситуациями?

7. Как вызывается функция abort (), что она выполняет, можно ли ее заменить своим кодом?

8. Для чего используют функции set\_unexpected () set\_terminate (), сколько они аргументов и возвращают?

Задание

1. Наберите текст программы приведенного примера обработки исключительных ситуаций и наладьте его (рассмотрите 2 случая работы программы). Объясните полученные результаты в первом случае и во втором.

2. Напишите программу, в которой пользователю предлагают несколько раз ввести число. Каждый раз выводить его корень. Если вводится отрицательное число, выводить сообщение об ошибке. Использовать технику исключительных ситуаций.

Задания для самостоятельной работы

Функции, которые реализуются в задачах обязательно должны выполнять проверку параметров, передаваемых и генерировать исключения в случаях ошибок.

1. Функция вычисляет площадь треугольника по трем сторонам S = sqrt (p (p - a) (p - b) (p - c)), где p = (a + b + c) / 2.

2.Функции вычисляет корень линейного уравнения ax + b = 0.

3.Функции вычисляет корень квадратного уравнения ax2 + bx + c = 0.

4.ФУНКЦИИ проверяет, является ли строка передаваемого полиндромом.