Лабораторная работа №4

"Обработка исключений"

Во всех заданиях при создании, чтении или записи файла использовать механизм генерации, контроля и обработки исключений.

1. Вручную сформировать текстовый файл, в который записать фамилии людей и номера их телефонов. Программно создать второй файл, в который записать фамилии, номера телефонов которых содержат не более заданного количества различных цифр. Во втором файле должны быть и фамилии, и номера телефонов.

2. Вручную сформировать текстовый файл, содержащий произвольные слова. Программно создать второй файл для записи всех слов, в которых количество различных символов составляет не менее половины от общего количества символов слова. Помимо самих слов записать долю различных символов от общего количества символов слова.

3. Вручную сформировать текстовый файл, содержащий фамилии студентов, количество экзаменов, которые сдавал каждый студент, и оценки, полученные на экзаменах. Программно создать второй файл для записи фамилий студентов, закончивших сессию без двоек, средний балл которых превышает средний балл по всей группе. При подсчете среднего балла, двойки во внимание не принимать.

4. Вручную сформировать текстовый файл, содержащий произвольные действительные числа. Программно создать второй файл для записи всех чисел, расположенных между максимальным и минимальным, или между минимальным и максимальным числами исходного файла. Числа в исходном файле должны быть перечислены в случайном порядке.

5. Вручную сформировать текстовый файл, содержащий произвольные слова. Найти в файле слова с наибольшей и наименьшей долей гласных букв. Программно создать второй файл для записи всех слов, расположенных между найденными словами. Доля гласных букв в слове определяется отношением количества гласных букв в слове к общему количеству букв в слове.

6. Вручную сформировать текстовый файл, содержащий произвольные слова. Подсчитать долю различных символов от общего количества символов слова. Программно создать второй файл для записи всех слов, в которых доля различных символов меньше среднего значения этой величины для всех слов файла. Во втором файле для каждого слова указать долю различных символов, расположенных между найденными словами.

Теоретическая справка к заданию

«ИСПОЛЬЗОВАНИЕ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ C++ ДЛЯ ОБРАБОТКИ ОШИБОК»

После того как вы создали и отладили (удалили ошибки) несколько программ, вы уже способны предвидеть ошибки, которые могут встретиться в программе. Например, если ваша программа читает информацию из файла, ей необходимо проверить, существует ли файл и может ли программа его открыть. Аналогично, если ваша программа использует оператор new для выделения памяти, ей необходимо проверить и отреагировать на возможное отсутствие памяти. По мере увеличения размера и сложности ваших программ вы обнаружите, что необходимо включить много таких проверок по всей программе. Из теоретической справки вы узнаете, как использовать исключительные ситуации C++ для упрощения проверки и обработки ошибок. После изучения материала вы усвоите следующие концепции:

Исключительная ситуация (exception) представляет собой неожиданное событие — ошибку — в программе.

В ваших программах вы определяете исключительные ситуации как классы.

Чтобы заставить ваши программы следить за исключительными ситуациями, необходимо использовать оператор C++ try.

Для обнаружения определенной исключительной ситуации ваши программы используют оператор C++ catch.

Для генерации исключительной ситуации при возникновении ошибки ваши программы используют оператор C++ throw.

Если ваша программа обнаруживает исключительную ситуацию, она вызывает специальную (характерную для данной исключительной ситуации) функцию, которая называется обработчиком исключительной ситуации.

Некоторые (старые) компиляторы не поддерживают исключительные ситуации C++.

C++ ПРЕДСТАВЛЯЕТ ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ КАК КЛАССЫ

Ваша цель при использовании исключительных ситуаций C++ состоит в упрощении обнаружения и обработки ошибок в программах. В идеале, если ваши программы обнаруживают неожиданную ошибку (исключительную ситуацию), им следует разумным образом ее обработать вместо того, чтобы просто прекратить выполнение.

В программах вы определяете каждую исключительную ситуацию как класс. Например, следующие ситуации определяют три исключительные ситуации для работы с файлами:

```
class file_open_error {};

class file_read_error {};

class file_write_error {};
```

Позже в этом уроке вы создадите исключительные ситуации, которые используют переменные и функции-элементы класса. А пока просто поверьте, что каждая исключительная ситуация соответствует классу.

КАК ЗАСТАВИТЬ C++ ПРОВЕРЯТЬ ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ

Прежде чем ваши программы могут обнаружить и отреагировать на исключительную ситуацию, вам следует использовать оператор C++ try для разрешения обнаружения исключительной ситуации. Например, следующий оператор try разрешает обнаружение исключительной ситуации для вызова функции file_copy:

```
try
{
  file_copy("SOURCE.TXT", "TARGET.TXT");
};
```

Сразу же за оператором try ваша программа должна разместить один или несколько операторов catch, чтобы определить, какая исключительная ситуация имела место (если она вообще была):

```
try
{
  file_copy("SOURCE.TXT", "TARGET.TXT");
};
catch (file_open_error)
{
  cerr << "Ошибка открытия исходного или целевого файла" << endl;
  exit(1);
}
catch (file_read_error)
{
  cerr << "Ошибка чтения исходного файла" << endl;
  exit(1);
}
catch (file_write_error)
{
  cerr << "Ошибка записи целевого файла" << endl;
  exit(1);
}
```

Как видите, приведенный код проверяет возникновение исключительных ситуаций работы с файлами, определенных ранее. В данном случае независимо от типа ошибки код просто выводит сообщение и завершает программу. В идеале ваш код мог бы отреагировать и не так — возможно, попытаться исключить причину ошибки и повторить операцию. Если вызов функции прошел успешно и исключительная ситуация не выявлена, C++ просто игнорирует операторы catch.

ИСПОЛЬЗОВАНИЕ ОПЕРАТОРА throw ДЛЯ ГЕНЕРАЦИИ ИСКЛЮЧИТЕЛЬНОЙ СИТУАЦИИ

Сам C++ не генерирует исключительные ситуации. Их генерируют ваши программы, используя оператор C++ throw. Например, внутри функции file_copy программа может проверить условие возникновения ошибки и сгенерировать исключительную ситуацию:

```
void file_copy(char *source, char *target)
{
  char line[256];
  ifstream input_file(source);
  ofstream output_file(target);
  if (input_file.fail())
     throw(file_open_error);
  else if (output_file.fail())
     throw(file_open_error);
  else
  {
    while ((! input_file.eof()) && (! input_file.fail()))
    {
      input_file.getline(line, sizeof(line)) ;
      if (! input_file.fail()) output_file << line << endl;
      else throw(file_read_error);
      if (output_file.fail()) throw (file_write_error) ;
    }
  }
}
```

Как видите, программа использует оператор throw для генерации определенных исключительных ситуаций.

Как работают исключительные ситуации

Когда вы используете исключительные ситуации, ваша программа проверяет условие возникновения ошибки и, если необходимо, генерирует исключительную ситуацию, используя оператор throw. Когда C++ встречает операторthrow, он активизирует соответствующий обработчик исключительной ситуации (функцию, чьи операторы вы определили в классе исключительной ситуации). После завершения функции обработки исключительной ситуации C++ возвращает управление первому оператору, который следует за оператором try, разрешившим обнаружение исключительной ситуации. Далее, используя операторысatch, ваша программа может определить, какая именно исключительная ситуация возникла, и отреагировать соответствующим образом.

ОПРЕДЕЛЕНИЕ ОБРАБОТЧИКА ИСКЛЮЧИТЕЛЬНОЙ СИТУАЦИИ

Когда ваша программа генерирует исключительную ситуацию, C++ запускает обработчик исключительной ситуации (функцию), чьи операторы вы определили в классе исключительной ситуации. Например, следующий класс исключительной ситуации nuke_meltdown определяет операторы обработчика исключительной ситуации в функции nuke_meltdown:

```
class nuke_meltdown
{
public:
  nuke_meltdown(void) { cerr << "\a\a\aРаботаю! Работаю! Работаю!" << endl; }
};
```

В данном случае, когда программа сгенерирует исключительную ситуацию nuke_meltdown, C++ запустит операторы функцииnuke_meltdown, прежде чем возвратит управление первому оператору, следующему за оператором try, разрешающему обнаружение исключительной ситуации. Следующая программа MELTDOWN.CPP иллюстрирует использование функции nuke_meltdown. Эта программа использует оператор try для разрешения обнаружения исключительной ситуации. Далее программа вызывает функцию add_u232 с параметромamount. Если значение этого параметра меньше 255, функция выполняется успешно. Если же значение параметра превышает 255, функция генерирует исключительную ситуацию nuke_meltdown:

```
#include <iostream.h>
class nuke_meltdown{
  public:
    nuke_meltdown(void) { cerr << "\a\a\aРаботаю! Работаю! Работаю!" << endl; }
};
void add_u232(int amount){
  if (amount < 255) cout << "Параметр add_u232 в порядке" << endl;
  else throw nuke_meltdown();
}
void main(void){
  try
  {
    add_u232(255);
  }
  catch (nuke_meltdown)
  {
    cerr << "Программа устойчива" << endl;
  }
}
```

Если вы откомпилируете и запустите эту программу, на экране дисплея появится следующий вывод:

C:\> MELTDOWN <ENTER>

Работаю! Работаю! Работаю!

Программа устойчива

Если вы проследите исходный код, который генерирует каждое из сообщений, то сможете убедиться, что поток управления при возникновении исключительной ситуации проходит в обработчик исключительной ситуации и обратно к оператору catch. Так, первая строка вывода генерируется обработчиком исключительной ситуации, т.е. функцией nuke_meltdown. Вторая строка вывода генерируется в операторecatch, который обнаружил исключительную ситуацию.

Определение обработчика исключительной ситуации

Когда C++ обнаруживает в программе исключительную ситуацию, он запускает специальную функцию, которая называется обработчиком исключительной ситуации. Для определения обработчика исключительной ситуации вы просто создаете функцию в классе исключительной ситуации, которая имеет такое же имя, как и сама исключительная ситуация (подобно конструктору). Когда ваша программа в дальнейшем сгенерирует исключительную ситуацию, C++ автоматически вызовет соответствующий обработчик. В идеале обработчик исключительной ситуации должен выполнить операции, которые бы исправили ошибку, чтобы ваша программа могла повторить операцию, ставшую причиной ошибки. После завершения обработки исключительной ситуации выполнение вашей программы продолжается с первого оператора, следующего за оператором try, разрешившего обнаружение исключительной ситуации.

ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТОВ ДАННЫХ ИСКЛЮЧИТЕЛЬНОЙ СИТУАЦИИ

В предыдущих примерах ваши программы, используя оператор catch, могли определить, какая именно исключительная ситуация имела место, и отреагировать соответствующим образом. В идеале, чем больше информации об исключительной ситуации могут получить ваши программы, тем лучше они смогут отреагировать на ошибку. Например, в случае с исключительной ситуацией file_open_error вашей программе необходимо знать имя файла, который вызвал ошибку. Аналогично, для файловых исключительных ситуаций file_read_error или file_write_error программа, возможно, захочет узнать расположение байта, на котором произошла ошибка. Чтобы сохранить подобную информацию об исключительной ситуации, ваши программы могут просто добавить элементы данных в класс исключительной ситуации. Если в дальнейшем ваша программа сгенерирует исключительную ситуацию, она передаст эту информацию функции обработки исключительной ситуации в качестве параметра, как показано ниже:

```
throw file_open_error(source);
throw file_read_error(344);
```

В обработчике исключительной ситуации эти параметры могут быть присвоены соответствующим переменным класса (очень похоже на конструктор). Например, следующие операторы изменяют исключительную ситуацию file_open_error, чтобы присвоить имя файла, который вызвал ошибку, соответствующей переменной класса:

```
class file_open_error{
public:
  file_open_error(char *filename) {strcpy(file_open_error::filename, filename); }
  char filename[255] ;
};
```

ОБРАБОТКА НЕОЖИДАННЫХ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Из урока 11 вы узнали, что компиляторы C++ предоставляют функции библиотеки этапа выполнения, которые вы можете использовать в своих программах. Если вы читали документацию по этим функциям, то могли обратить внимание на функции, которые генерируют определенные исключительные ситуации. В таких случаях ваши программы должны проверять соответствующие исключительные ситуации. По умолчанию если ваша программа генерирует исключительную ситуацию, которая не улавливается (программа не имеет соответствующего обработчика исключительной ситуации), то запустится стандартный обработчик, предоставляемый языком C++. В большинстве случаев стандартный обработчик завершит вашу программу. Следующая программа UNCAUGHT.CPP иллюстрирует, как стандартный обработчик исключительной ситуации завершает выполнение вашей программы:

```
#include <iostream.h>
class some_exception { };
void main(void)
{
  cout << "Перед генерацией исключительной ситуации" << endl;
  throw some_exception();
  cout << "Исключительная ситуация сгенерирована" << endl;
}
```

В данном случае, когда программа генерирует исключительную ситуацию (которая не улавливается программой), C++ вызывает стандартный обработчик исключительной ситуации, который завершает программу. Поэтому последний оператор программы, который выводит сообщение о генерации исключительной ситуации, никогда не выполняется. Вместо использования стандартного обработчика исключительной ситуации C++ ваши программы могут определить свой собственный стандартный обработчик (обработчик по умолчанию). Чтобы сообщить компилятору C++ о своем стандартном обработчике, ваши программы должны использовать функцию библиотеки этапа выполнения set_unexpected. Прототип функции set_unexpected определен в заголовочном файлеexcept.h.

ОБЪЯВЛЕНИЕ ГЕНЕРИРУЕМЫХ ФУНКЦИЕЙ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Как вы уже знаете, прототип функции позволяет вам определить тип возвращаемого функцией значения и типы ее параметров. Если ваши программы используют исключительные ситуации, вы также можете использовать прототип функции, чтобы указать генерируемые данной функцией исключительные ситуации. Например, следующий прототип функции сообщает компилятору, что функция power_plant может генерировать исключительные ситуации melt_down и radiation_leak:

```
void power_plant(long power_needed) throw (melt_down, radiation_leak);
```

Включая подобным образом возможные исключительные ситуации в прототип функции, вы можете легко сообщить другому программисту, который будет читать ваш код, какие исключительные ситуации ему необходимо проверять при использовании данной функции.

ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ И КЛАССЫ

При создании класса вы, возможно, захотите определить исключительные ситуации, характерные для данного класса. Чтобы создать исключительную ситуацию, характерную для конкретного класса, просто включите эту исключительную ситуацию в качестве одного из общих(public) элементов класса. Например, следующее описание класса string определяет две исключительные ситуации:

```
class string
{
public:
  string(char *str);
  void fill_string(*str);
  void show_string(void);
  int string_length(void);
  class string_empty { } ;
  class string_overflow {};
private:
  int length;
  char string[255];
};
```

Как видите, этот класс определяет исключительные ситуации string_empty и string_overflow. В своей программе вы можете проверить наличие исключительной ситуации, используя оператор глобального разрешения и имя класса, как показано ниже:

```
try
{
  some_string.fill_string(some_long_string);
};
catch (string::string_overflow)
{
  cerr << "Превышена длина строки, символы отброшены" << endl;
}
```

ЧТО ВАМ НЕОБХОДИМО ЗНАТЬ

Исключительные ситуации предназначены для упрощения и усовершенствования обнаружения и обработки ошибочных ситуаций в ваших программах. Для проверки и обнаружения исключительных ситуаций ваши программы должны использовать операторы try, catch ithrow. Ваши знания исключительных ситуаций зависят от опыта программирования на C++. Прежде чем продолжить программировать на C++, убедитесь, что вы освоили следующие основные концепции:

Исключительная ситуация представляет собой неожиданную ошибку в вашей программе.

Ваши программы должны обнаруживать и обрабатывать (реагировать на) исключительные ситуации.

В программах вы определяете исключительную ситуацию как класс.

Вы должны использовать оператор try, чтобы заставить компилятор C++ разрешить обнаружение исключительных ситуаций.

Вы должны разместить оператор catch сразу же после оператора try, чтобы определить, какая именно исключительная ситуация имела место (если она вообще была).

C++ сам не генерирует исключительные ситуации. Ваши программы генерируют исключительные ситуации с помощью оператора throw.

Если ваша программа обнаруживает исключительную ситуацию, она вызывает специальную функцию, которая называется обработчиком исключительной ситуации.

Если ваши программы используют исключительные ситуации, вы можете указать в прототипе функции, что эта функция способна генерировать исключительную ситуацию.

При чтении документации по библиотеке этапа выполнения обращайте внимание на то, что некоторые функции способны генерировать исключительные ситуации.

Если ваша программа генерирует, но не улавливает (и не обрабатывает) исключительную ситуацию, C++ будет вызывать стандартный обработчик исключительной ситуации.

В заголовочном файле except.h определены прототипы функций, которые ваши программы могут использовать для указания своих собственных стандартных обработчиков не улавливаемых исключительных ситуаций.