Лабораторная работа № 6

Обработка исключений.

Цель работы: Формирование навыка обработки ошибок в виде исключений, повышающих надёжность создаваемых программ. Формирование навыка создания проекта по техническим требованиям. Закрепление навыков работы с классами.

Теория

Развитый механизм обработки ошибок – это надёжный путь к повышению надёжности программ.На практике программисты часто игнорируют ошибки (например возвращает количество успешно выведенных символов, однако никто не проверяет результат её вызова – это увеличение кода программы). В классическом Си вызов функции тесно связан с обработкой ошибок, код обработки «перемешан» с основным кодом. В C++ механизм обработки исключений позволяет сначала описать нормальное течение вычислений, а затем, в отдельной секции, описать код для решения различных проблем. Таким образом при многократном вызове функции обработка ошибок этой функции производится в одном и том же месте.
Обработка исключений – это системные средства для обработки ошибок времени выполнения.

Инструкции try, catch, throw образуют взаимосвязанную подсистему

Общие правила:

1. throw генерирует (выбрасывает) исключение, которое перехватывается catch-инструкцией (обработчик ошибок)
2. try-блок определяет отрезок программы, в котором может возникнуть исключение (т.е. где проверяются ошибки)
3. с try-блоком может быть связано несколько catch-инструкций
4. какой именно catch- блок будет выполняться определяет типом исключения (встроенный или пользовательский)
5. Инструкция throw exception должна выполняться либо в блоке try, либо в функции вызванной из блока try
6. Если генерируекся throw exception для колорого нет обработчика (catch-блока), произойдет аварийное завершение программы terminate()

По-умолчанию выполняется abort(), но реакцию можно изменить.

Пример:

```
void main()
{  cout<<"Begin\n";
        try {
        cout<<"in try block\n";
    throw 99; // генерация int-исключения
    cout<<"do not work!!!";
 }
// Обработчик ошибок
catch (double k)
{cout<<"catch double exception:"
<< k << "\n" ;
}
catch (int i)
```

```
{cout<<"catch int exception:"
<< i << "\n" ;
}
cout<<"The end"<<endl;
system ("pause");
return 0;
}
```

Результат работы программы :

```
Begin
in try block
catch int exception: 99
```

Перехват исключений классового типа

Большинство реальных программ генерируют исключения типа класс

Пример:

```
class except
{public:
  char msg[80];
  except()              {*msg=0;}
  except(char* s)  {strcpy(msg,s);}
};  //------- end except

double fdiv (double a, double b)
{if (b)
{       cout<<"+++"<<endl;
        return (a/b);}
else throw except("error: divider=0");
}

double fsqrt (double q)
{if (q<0) throw except("error: sqrt");
return sqrt(q);
}

double fasin (double r)
{if (r<-1 || r>1)
throw except("error fasin");
return asin(r);
}


void main()
{ double x1,y1,z1;
        cout<<"x1="; cin>>x1;
        cout<<"y1="; cin>>y1;
try
{cout<<"z1="<<fdiv (x1,y1)<<endl;
  cout<<"z2="<<fsqrt (x1)<<endl;
  cout<<"z3="<<fasin (y1)<<endl;
}
catch (except e)
{cout<<e.msg<<endl;
```

```
}
cout<<”end”;
}
```

Перехват всех исключений

Если различные исключения обрабатываются одинаково, то используется обработчик исключений вида catch (...)
Следующий пример показываент эту ситуацию. В функции capture () могут генерироваться исключения различного типа, которые обрабатываются одним блоком catch (...)

Пример:

```
void capture (int k)
{try    {
    switch (k) {
        case 1 :  throw 1;
        case 2:   throw 'a';
        case 3 :  throw 2.8;
                }
        }

catch (...) // перехват всех исключений
{cout<<"all type capture\n";}
}
void main()
{  cout<<"Begin\n";
        capture (1);
        capture (2);
        capture (3);
        cout<<"The end"<<endl;
}
```

Результат на экране :

```
Begin
all type capture
all type capture
all type capture
The end
```

Если в наличии несколько catch – инструкций, и catch (...) последняя из них, то будет перехват «всех остальных»

Пример:

```
void capture (int k)
{try    {
    switch (k) {
        case 1 :  throw 1;
        case 2:   throw 'a';
        case 3 :  throw 2.8;
        case 4 :  throw "string";
       case 5 :  throw except("error!!!");                }
        }
```

```
catch (int i) {
cout<<"catch int exception:" <<i<<endl;}
catch (char s) {
cout<<"catch char exception:" <<s<<endl;}
 catch (except e)
{cout<<e.msg<<endl;
}
catch (...) // перехват всех остальных исключений
{cout<<"all type capture\n";}
}

void main()
{  cout<<"Begin\n";
capture (1);
capture (2);
capture (3);
capture (4);
capture (5);
cout<<"The end"<<endl;
}
```

Результат на экране :

```
Begin
catch int exception: 1
catch int exception: a
all type capture
all type capture
error!!!
The end
```

Общие правила и требования

1. При оформлении ввода-вывода данных информация на экране должна быть отформатирована, согласно требованиям предыдущих лабораторных работ
2. В данной лабораторной работе вы учитесь создавать проект без точных указаний по реализации базы данных. Такая постановка задачи называется работа по техническому заданию, именно так работает программист в реальных условиях.
3. В задании указан набор обязательных элементов проекта, но при реализации взаимодействия классов от вас потребуется добавить элементы:
   - необходимые для работы (счетчики, размеры массивов и т.п.);
   - не обязательные, но упрощающие реализацию алгоритмов обработки данных (для промежуточных данных, для хранения общих данных и т.п.)
   - вы также можете усложнить иерархию и расширить интерфейсы в рамках поставленной задачи (это добавит вам баллы при защите работы).
4. Общие требования к проекту:
   - Данные во всех классах должны быть защищенные (private или protected).
   - Обработка любых ошибок должна реализовываться в виде исключений
   - Данные, вносимые по ходу работы программы должны вводиться с клавиатуры, предусмотреть контроль корректности ввода и обработку ошибки. Например:
     — контроль диапазона значений: 1-12(для месяца), 0-31 (для дня,)0-24(для часа), и т.п.;
     — обработка ошибок при вычислениях, например запрет деления на ноль и т.п.

5. В задании ничего не говориться о выводе информации на экран, но от решения этого вопроса зависит качество проекта в целом. В целом вывод можно разделить на три категории:
   - информация выводится по умолчанию при запуске программы;
   - информация выводится по запросу пользователя (пункт меню);
   - вывод информации – это результат выполнения какой-либо операции.

   Обычно требуется использовать все три категории вывода. Должен быть пункт меню «распечатать всю информацию».

Номер компьютера	Варианты заданий
3,13, 23	**При выполнении задания используйте общие требования выполнения работы** Добавить новые элементы к базе данных Склад (товары) 1. Перегрузить оператор ввод из входного потока cin «**>>**» для классов **storage,stock** 2. Использовать только оператор >> для ввода с клавиатуры (внести необходимые исправления в текст программы) 3. Добавить обработку возможных ошибок во всех случаях ввода информации с клавиатуры, там, где это возможно предусмотреть режим «перехват всех остальных исключений». 4. Добавить действие: приход товара на склад. Если название и цена совпадают, то добавить количество, а если нет, то добавить новую позицию склада 5. Добавить действие: расход со склада. Обработать ошибку при попытке списания большего количества, чем есть на складе.