// Programming Techniques
Exercises Class # 4

Master Programme in Mathematical Finance
1st Semester 2018/2019
ISEG-UL

Sara Lopes
{
//sblopes@iseg.ulisboa.pt
//saradutralopes@gmail.com
}

Overview

- Errors
  - Sources of errors
  - Compile-time errors
  - Link-time errors
  - Run-time errors
  - Exceptions
- Debugging
- Function Pointers
- File Organization

Bibliography:

- Stroustrup, Bjarne, Programming Principles and Practice Using C++, Second Edition, Addison-Wesley, 2014.

Classification of Errors

- **Compile-time errors** (syntax errors; type errors): errors found by the compiler;
- **Link-time errors**: errors found by the linker when it is trying to combine object files into an executable program
- **Run-time errors**: Errors found by checks in a running program
- **Logic errors**: Errors found by the programmer looking for the causes of erroneous results

Your program:

1. Should produce the desired results for all legal inputs
2. Should give reasonable error messages for all illegal inputs

Sources of Errors

- Poor specification
- Incomplete programs
- Unexpected arguments
- Unexpected input
- Unexpected state
- Logical errors

Compile-time Errors

The compiler is your first line of defense against errors - the compiler analyzes the code to detect syntax errors and type errors.

Syntax Errors: commands not well formed according to the C++ grammar

```
int area(int length, int width); // calculate area of a rectangle

int s1 = area(7; // error: ) missing
int s2 = area(7) // error: ; missing
Int s3 = area(7); // error: Int is not a type
int s4 = area('7); // error: non-terminated character
```

Compile-time Errors

Type Errors: mismatches between the types you declared for your variables and the types of values or expressions you assign to them.

```
int area(int length, int width); // calculate area of a rectangle

int x1 = area(7); // error: wrong number of arguments
int x2 = area("seven",2); // error: 1st argument has a wrong type
```

Link-time errors

- Every function in a program must be declared with exactly the same type in every translation unit in which it is used
- Every function must also be defined exactly once in a program

If either of these rules is violated, the linker will give an error.

Run-time errors

```
int area(int length,int width)
{
        return length*width;
}

int framedarea(int x,int y)
{
        return area(x-2,y-2);
}
....
int x=-1;
int y=2;
int z=4;

int area1=framedarea(3,z);//result=2
int area2=framedarea(x,y);//result=0
double ratio=double(area1)/area2;// Division by 0!
```

How to deal with errors?

Run-time errors

Checking for valid arguments inside the function.

```
int framedarea(int x,int y)
{
 const int frame_width=2;
 if(x-frame_width<=0 || y-frame_width<=0)
  error("non-positive area() argument called by framed_area()");
 return area(x-2,y-2);
}

int area(int length, int width) // calculate area of a rectangle
{
 if (length<=0 || width <=0) error("non-positive area() argument")
 return length*width;
}
```

Check your arguments in a function unless you have a good reason not to.

Exceptions

Separate detection of an error from the handling of an error while ensuring that a detected error cannot be ignored.

```
class Bad_Area{};//type specific for reporting errors from area()

int area(int length, int width)
{
 if (length<=0 || width <=0) throw Bad_Area{};
 return length*width;
}

int main()
{
 try{
  ...
  int z=area(x,y);
 }
 catch(Bad_Area){
  cerr<< "Bad Area arguments"<<endl;
 }
 return 0;
}
```

Vector range errors

```
int main()
{
 try{
   vector<int>v;
   int x;
   while(cin>>x) v.push_back(x);
   for(int i=0;i<=v.size();i++)
    cout<<"v["<<i<<"]=="<<v[i]<<endl;
 }
 catch(out_of_range_error){
  cerr<< "Range error"<<endl;
  return 1;
 }
 catch(...){ //catch all other exceptions
  cerr<< "Some error happened"<<endl;
  return 2;
 }
 return 0;
}
```

Input errors

```
int main(){
 cout<< "type a value between 1 and 10\n";
 int x = -1;
 cin >> x;

 if (!cin)
  error("error in value input");

 if (x < 1 || 10 < x)
  error("x is out of range");

 return 0;
}
```

error() is supposed to terminate the program after getting its message written

Input errors

…but we might want to take some minor action before exiting. We do that with exceptions.

```
int main(){
 try{
  cout<< "type a value between 1 and 10\n";
  int x = -1;
  cin >> x;
  if (!cin)
   error("error in value input");
  if (x < 1 || 10 < x)
   error("x is out of range");
}
 catch(runtime_error& e){
  cerr<<"runtime error "<<e.what()<<endl;
 return 1;
 }
 return 0;
 }
```

Debugging

1. Get the program to work
2. Get the program to link
3. Get the program to do what is supposed to do

and each time time something doesn't work we have to find what caused the problem and fix it.

Practical debug advice

1. Comment your code well
2. Use meaningful names
3. Use a consistent layout of code
4. Break code into small functions
5. Avoid complicated code sequences
6. Try to avoid nested loops, nested if-statements, complicated conditions...
7. Use library facilities rather than your own code when you can.

Function Pointers

```
#include"std_lib_facilities.h"

double f1(int i, double d){
return i*d;
}
double f2(int i, double d){
return i+d;
}
int main() {
double (*fptr)(int,double);
//declares fptr as a pointer to a function
//that receives an int and a double and returns a double
fptr=&f1;
cout<<fptr(1,1.2)<<endl;
fptr=&f2;
cout<<fptr(1,1.2)<<endl;
return 0;
}
```

Numerical Integration

If we want to integrate a function using the Rectangle Method:

$$\int_a^b f(x)dx \approx h \sum_{i=0}^{N-1} f(x_n)$$

where $h = (b - a)/N$ and $x_n = a + ih$.

```
#include"std_lib_facilities.h"
typedef double (*fptr)(double);

double integral(fptr f, double a, double b,double h){
int n=(b-a)/h;
double sum=0;
for(int i=0;i<n;i++){
sum+=h*f(a+i*h);
}
return sum;
}
double f1(double x){
return x*x;
}

int main() {
fptr f=f1;
cout<<Integral(f,0,1,0.1)<<endl;
return 0;
}
```

File Organization

A program is a set of functions organizes in one or more files that communicate between them.

func.h

```
#include<vector>
using namespace std;

//some definitions
void print(vector<double> );
double multiply(vector<double> ,vector<double> );
....
```

func.cpp

```
#include"func.h"
#include"std_lib_facilities.h"

void print(vector<double> ){
for(size_t i=0;i<v.size();i++)
cout<<v[i]<<" ";
cout<<endl;
}
double multiply(vector<double>v,vector<double>u ){
double sum=0;
for(size_t i=0;i<v.size();i++)
sum+=v[i]*u[i];
return sum;
}
....
```

main.cpp

```
#include"func.h"
int main(){
vector<double>v(5,1);
vector<double>v(5,2);

print(v);
cout<<multiply(v,u)<<endl;

return 0;
}
```