// Programming Techniques

Exercises Class # 2

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

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

Overview

- Control Structures
  - Operator ?:
  - Switch
  - For
  - While
- Functions

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

Operator ?:

```
condition ?  expression 1 :  expression 2
```

If the condition is true the result is expression 1, else the result is the expression 2.

```
int main()
{
 int a,b,c;
 a = 2;
 b = 7;
 c = (a>b) ? a : b;
 cout<<c;
 return 0;
}
```

Result: **7**

Switch

```
int main()
{
 int x;
 cout<<" Choose an option(1,2,3):";
 in>>x;
 switch(x){
   case 1:
   cout<<"option 1 selected\n";
   break;
   case 2:
   cout<<"option 2 selected\n";
   break;
   case 3:
   cout<<"option 3 selected\n";
   break;
   default:
   cout<<"Invalid option";
   break;
  }
}
```

Switch

- The value on which we switch must be an **int**, **char** or **enumeration** type.
- The values in the case labels must be **constant expressions** (not variables)
- Two labels can't have the same value
- Always use **break** in the end of each case.

For

```
for(expression1; condition; expression2)[instructions]
```

It works in the following way:

1. expression 1 is executed. (usually a control variable is initialized),
2. the condition is evaluated. If it is false then the cycle ends, otherwise it continues,
3. the instruction is executed,
4. expression 2 is executed. (typically an increment or decrement of the control variable)
5. goes back to 2.

For

```cpp
int main()
{
 //countdown using a for loop

 for (int n=10;n>0;n--)
 {
 cout<<n<<" ";
 }
 return 0;
}
```

Result: 10 9 8 7 6 5 4 3 2 1

For

The flow of the execution of a for cycle can be altered using the instructions break and continue:

1. **break**: ends the execution of a cycle by jumping to the following instruction.
2. **continue**: jumps to instruction 2 of the cycle.

For

```
int main()
{
 //countdown using a for loop

 for (int n=10;n>0;n--)
 {
 if(n==5) continue;
 cout<<n<<" ";
 }
 return 0;
}
```

```
Result: 10 9 8 7 6 4 3 2 1
```

For

```
int main()
{
 //countdown using a for loop

 for (int n=10;n>0;n--)
 {
 if(n==5) break;
 cout<<n<<" ";
 }
}
```

```
Result: 10 9 8 7 6
```

While

```
while (condition) instruction
```

While condition is true execute the instruction.

```
int main()
{
 //countdown using a while loop
 int n=10;
 while(n>0)
 {
 cout<<n<<" ";
 n--;
 }
}
```

Result: **10 9 8 7 6 5 4 3 2 1**

Do While

`do instruction while (condition)`

Execute the instruction while condition is true.

```
int main()
{
 //countdown using a while loop
 int n=10;
 do
 {
 cout<<n<<" ";
 n--;
 }while(n>0);
}
```

Result: **10 9 8 7 6 5 4 3 2 1**

Functions

```
type identifier ([type argument1],[type argument2],...)
```

```
int addition (int a, int b)
{
 int r;
 r=a+b;
 return r;
}

int main()
{
 int z;
 z=addition(3,5);
 cout<<"The result is"<<z;
 return 0;
}
```

Functions

```cpp
int addition (int, int);//declaration

int main()
{
 int z;
 z=addition(3,5);
 cout<<"The result is"<<z;
 return 0;
}

int addition (int a, int b) //definition
{
 int r;
 r=a+b;
 return r;
}
```

Functions

A function that doesn't return any value is a **void** type function.

```
void printmessage ()
{
 cout<<"I'm a function \n";
}

int main()
{
 printmessage();
 return 0;
}
```

Functions

When we define a function we can choose to have default values for the last arguments.

```
double divide (double a, double b=1.0)
{
double d=a/b;
return d;
}

int main()
{
 cout<<divide(3)<<","<<divide(12,2);
 return 0;
}
```

Result: **3, 6**

Functions

We can define different functions with the same name if the list of arguments is different.

```
double operate(double a, double b)
{
return (a/b);
}

int operate(int a, int b)
{
return (a*b);
}
int main()
{
 int x=5,y=2;
 double n=5.0,m=2.0;
 cout<<operate(x,y)<<endl;
 cout<<operate(n,m)<<endl;
}
```

Result: **10, 2.5**

Functions

The arguments of a function can be passed on by **value** or **reference**.

By value: The value doesn't change outside the function.

By reference: In this case the function cn change the value in memory.

```

void duplicate(int& a)
{
a*=2;
}
int main()
{
 int a=3;
 duplicate(a);
 cout<<a<<endl;
}
```

Result: **6**

Function Recursivity

Recursion: when a function references itself.

```

long factorial(long a)
{
 if(a>1)
  return(a*factorial(a-1));
 else
  return(1);
}
int main()
{
 cout<<factorial(5)<<endl;
}
```

```
Result: 120
```

Exercises

- Write a function that returns the sum of the first n natural numbers.
- Calculate the product of two int numbers without using the * operator.