CSE 165/ENGR 140 Intro to Object Orient Program

Lecture 3 – C in C++

(Ch. 3)

Announcement

- Lab #1 this week
  - Due before next lab begins
- Reading assignment
  - Ch. 3 (Quizzes will be based on lecture and reading)

Functions

- A block of statements “called” from a program:

```
type name ( argument1, argument2, ...)
{
    statement1
    statement2
    …
}
```

Functions

- A function:
  - may return a value (use void if not returning a value)
  - may use and change arguments
  - can be called multiple times
  - could be reused in multiple programs
  - it enables to modularize the code (building blocks)

Functions

- Declaration

```
int translate ( float x, float y, float z );
int translate ( float, float, float );
```

- Definition

```
int translate ( float x, float y, float z )
{
    x = y = z;
}
```

- Functions of same name may have different arguments

Functions

- Example Definitions

```
int addIntegers ( int x, int y )
{
    return x + y;
}
void say ( string sentence)
{
    cout << sentence << endl;
}
```

- Calling the functions

```
int answer = addIntegers ( 5, 7);
say ( "Hello World!" );
```

Execution Control: if-else

```
// Demonstration of if and if-else conditionals
#include <iostream>
using namespace std;
int main() {
    int i;
    cout << "Type a number and hit 'Enter'" << endl;
    cin >> i;
    if (i > 5)
      cout << "It's greater than 5" << endl;
    else
        if (i < 5)
           cout << "It's less than 5 " << endl;
    else
        cout << "It's equal to 5 " << endl;
}
```

Execution Control: if-else

- ***To avoid obscure if-else chains, use { } and indentations.***

```
// Demonstration of if and if-else conditionals
#include <iostream>
using namespace std;
int main()
{
    int i;
    cout << "type a number and 'Enter'" << endl;
    cin >> i;
    if (i > 5)
        cout << "It's greater than 5" << endl;
    else
    {
        if (i < 5)
            cout << "It's less than 5 " << endl;
        else
            cout << "It's equal to 5 " << endl;
    }
}
```

Execution Control: while

while (<boolean_expression>)
    <loop body>

```
// Guess a number (demonstrates "while")
#include <iostream>
using namespace std;
int main()
{
  int secret = 15;
  int guess = 0;

  // "!=" is the "not-equal" conditional:
  while(guess != secret){ // Compound statement
    cout << "guess the number: ";
    cin >> guess;
  }

  cout << "You guessed it!" << endl;
}
```

Execution Control: do-while

do

    <loop body>

while (<boolean_expression>)

```
// The guess program using do-while
#include <iostream>
using namespace std;
int main() {

  int secret = 15;
  int guess; // No initialization needed here

  do {
    cout << "guess the number: ";
    cin >> guess; // Initialization happens
  } while ( guess!=secret );

  cout << "You got it!" << endl;
}
```

Execution Control: for

for ([<initialization>]; [<condition>]; [<increment>])
< loop body (statement; or {block}) >

```
// Display all the ASCII characters
// Demonstrates "for"
#include <iostream>
using namespace std;
int main()
{
  for (int i = 0; i < 128; i++)
   {
     if (i != 26) // ANSI Terminal Clear screen
        {
             cout << " value: " << i
                  << " character: "
                  << char(i) // Type conversion
                  << endl;
        }
   }
}
```

Wake up

- https://youtu.be/kKAkj3rCBEo

Execution Control: break/continue

- Break
  - Exit the loop
- Continue
  - Skip current iteration

```
// Simple menu demonstrating "break" and "continue"
#include <iostream>
using namespace std;
int main(){
  char c; // To hold response
  while(true){
    cout << "MAIN MENU> c: continue, q: quit -> ";
    cin >> c;
    if ( c == 'q' ) break; // Out of "while(1)"
    if ( c == 'c'){
       cout << "Press a or b: ";
       cin >> c;
       if ( c == 'a' ){
          cout << "you chose 'a'" << endl;
          continue; // Back to main menu
       }
       if ( c == 'b' ){
          cout << "you chose 'b'" << endl;
          continue; // Back to main menu
       }
     }
  }
}
```

Execution Control: switch

```
char command;
if (command == 'I')
    <statement1>
else if (command == 'R')
    <statement2>
else
    <statement3>
```

```
char command;
…
switch (command)
{
    case 'I' :
        <statement1>
        break;
    case 'R' :
        <statement2>
        break;
    default :
        <statement3>
        break;
}
```

Execution Control: switch

```
// A menu using a switch statement
#include <iostream>
using namespace std;
int main(){
  bool quit = false; // Flag for quitting
  while(!quit){
    cout << "Select a, b or q to quit: ";
    char response;
    cin >> response;
    switch(response){
      case 'a' : cout << "you chose 'a'" << endl;
                 break;
      case 'b' : cout << "you chose 'b'" << endl;
                 break;
      case 'q' : cout << "quitting menu" << endl;
                 quit = true;
                 break;
      default : cout << "Please use a,b, or q!" << endl;
    }
  }
}
```

Execution Control: goto

- Good programming style avoids using ***goto***

```
//: C03:gotoKeyword.cpp
// The infamous goto is supported in C++
#include <iostream>
using namespace std;
int main() {
  long val=0;
  for ( int i=1; i<1000; i++ ) {
    for ( int j=1; j<100; j+=10 ) {
      val = i * j;
      if ( val>47000 )
        goto bottom;
        // break would only go to the outer 'for'
        // use of goto may be justified in such cases
    }
  }
  bottom: // A label
  cout << val << endl;
}
```

Data types: scope

- Variables can be defined anywhere
- Scope of a variable
  - global:
    - Can be used through out the program
    - Use extern keyword for variables used across multiple files
  - local: within a scope
  - global but of limited scope: use static keyword
    - Limited to a file
    - More on static in future lectures

Data types: primitive types

- Types:
  - char, int, float, double
- Modifiers:
  - unsigned, signed, short, long
- Type limits:
  - Data type size varies depending on systems (16- vs 32- vs 64-bit CPU)
  - Use ***sizeof*** operator to determine size of each data type

Data types: bool

- ***bool*** was introduced in C++
- Can only contain true (1) or false (0)
- Conditional expressions always produce boolean types

Data types: bool

Examples:

```
bool is_small = a<=10;
cout << (is_small? "small" : "large");
```

```
bool in_unit_interval = n>=0.0 && a<=1.0? true:false;
```

```
bool initialized = false;
if(!initilized)
      init();
```

```
bool newcmd = true;
while(newcmd)
 {
      newcmd = enter_new_command();
 }
```

Data types: constants

- Modifier ***const***:
  - Can be of any type and in any scope
  - Always has to be initialized
    - const int x = 10;
  - Macro (pre-processor) alternative: # define PI 3.14159
    - Use ***const*** whenever possible
- More Complex Macros:

```
# define GS_ABS(x) ((x)>0? (x):-(x))
# define GS_TODEG(r) (180.0f*float(r)/gspi)
# define GS_MIN(a,b) ((a)<(b)? (a):(b))
# define GS_MIN3(a,b,c) ((a)<(b)?
        ((a)<(c)?(a):(c)):((b)<(c)?(b):(c)))
```