C/C++ Part 2 – Object Oriented programming with C++

Lecture 2

Outline

- Programming Template
- Input and Output Streams
  - the stream model
  - stream state
- Constructions
  - sequencing
  - selections
  - iteration

Code Template – C++

```
// import common i/o declarations
// e.g. cin and cout
#include <iostream>
// expose c in / cout from standard namespace
using std :: cin ;
using std :: cout ;
// program ent r y p oi n t
int main ( int argc , char* argv [ ] )
{
/ / write C++ code here
}
```

- Using sizeof with fundamental types
  - Load sizing.cpp

C++ Input & Output

- C++ has no language primitives (keywords) for I/O communication between the program and the environment occurs via a library of objects and associated functions
- The objects are global channels called (streams)
- Streams are library objects and are declared e.g. named and typed in <iostream>
- The streams are named cout for output and cin for input
- Special functions (with streams and types) called operators used to send/receive data to/from streams:
  - Output (cout) uses insertion operator << this inserts data from the program onto the output stream: cout << 42 << "Hello World" ;
  - Input (cin) pulls data with extraction operator >> off stream into the program: cin >> aNumber ;

Stream model

- The IO Stream Library is a hierarchy of components for buffered and non-buffered I/O of text and numerical values
- The stream has two ends, one within the program, the other to the environment - attached to a peripheral (e.g. keyboard, modem, printer, network connection or connected serial or USB port)
- The stream manages bytes inserted onto or extracted from the peripheral to the program
  - In input, the extraction end is the terminal or disk file, while the destination is a program variable

    (src - peripheral:) cin >> (dst - program:) var
  - In output, the insertion end is a program variable or literal, the destination is file, port or printer

    (dst - peripheral:) cout << (src - program:) var
- Buffers are used to mediate different throughput rates

Terminal I/O - Input

```
//  the channel to the left of the operator
// the data on the right.
int myint;  // no init as about to be read
cin >> myint ;    // read  integer from stream
```

Terminal I/O - Output

```
// the channel to the left o f the operator
// the data on the right.
int myint {42};// C++ 11 init     syntax
cout << myint;//write integer to stream
```

State and Format

- A stream has different characteristics including a state and formatting settings:
  - state includes connection with other streams and stdio, modes, error handling and locales
  - formatting - depends on state, locale, object type and manipulatory operations (manips) - and includes presentation and use of integer bases (8,10,16), floating point formatting and precision, processing of whitespace, fill, alignment and padding

State functions

- Stream can be tested using member functions . . .
  - fail() extraction failed to read expected value or insertion failed to write expected value
  - bad() means stream is corrupted (file read error?)
  - eof() means end of file is reached
  - good() means all bits are clear, stream okay for use

Activity 1

- I/O terminal stream processing
  - Load stream.cpp

- Practice
  - Write a program to print out a menu of months where 1=January and 12=December

Selection / Iteration

- Normally the order of execution of C++ code is the order listed in sequence in the source code.
- Sequencing is our first programming construct:

```
a = 42; //line 1 - do this first...
b = 96; //line 2 - ... then this...
c = a + b; //line 3 - ... and finally
```

- There are two other programming constructs in C++
  - Selection (branching) and
  - Iteration (looping)
- A parallel construct exists in some languages (e.g. OCCAM) for parallel CPU

Selection

- Execution of program follow line order in source code (top to bottom and left to right)
- Altering flow of code can be conditional evaluating then execution or unconditional where execution happens regardless
- Selection constructs allow programmer to chose which block of code to run
- Conditional testing for the truth or falsehood of a specific expression - normally at the top of the construct
  - A true expression is bool true and falsehood is represented by bool false
  - Remember that conversions or promotions can be used within the control test
  - Falsehood is a zero value, truth is !0
- The unconditional constructions are goto, return, break and continue

IF example

```
int result{0}
cout<<“Enter result:”;
cin>>result;
if(result>80) {
     cout<<“Distinction”;
} else if(result>=45)
        {cout<<“Pass”;
} else {
//neither pass nor distinction
     cout<<“Fail”;
}
```

SWITCH Example

```
char answer{‘ ’};
cout << “Do you wish to continue (y/n)?”;
cin >> answer ;
switch (answer) {
      case ’y’:cout<< "YES, TO CONTINUE”; break ;
      case ’n’:cout << "NO, TO EXIT " ; break ;
      default :cout << "Unknown Response " ;
      // default option ?
}
```

Conditional Operator

```
int result { 0 } ;
// get result
cout << "The result is "
     << (result <= 50 ? "Bad" : "Good" ) ;
```

Selection Summary

- Note that for the switch statement the condition must be a integral, enumeration or class with a conversion to integral / enumeration exists
- The case value should be a constant that can be promoted to the condition type
- Flow control is implemented via a break within the case code block
- The switch statement tends to easier to read than multiple arms of an if...else block
- The conditional expression (? :) is an expression and can be used different code locations, however it only provides a binary choice
- Nested conditional expressions are possible.

- EXAMPLE: Load selections.cpp

Iterations

- The final programming construction type, is those used for iteration (looping) over blocks of code
- there are three requirements for successful execution of a loop
  - initial statements - preparing the loop testing and calculating variables
  - conditional values or expression - for evaluating if the loop should continue or finish (loop control variable)
  - body statements - a sequence of code that performs the work of the loop, may also modify loop control variable
- C++ contains three iteration constructs for repeatedly executing sequences of code: for, while and do while

For loop

```
int daysInWeek {7} ;
for (int i=1; i<=daysInWeek; i ++ )
{
    cout<<"Day : “<< i << ‘\ n’ ;
} //print 1to7 inclusive
```

- Programming solution

  LOAD: forpos.cpp

While Loop

```
int daysInWeek {7};
int i {0};
while ( i <= daysInWeek ) {
     cout << i++ << '\ n';
}
```

- Programming solution
  - LOAD whilepos.cpp

Do While Loop

```
int daysInWeek {7} ;
int i {0};
do{
     cout << i++ << '\n';
} while ( i <= daysInWeek );
```

- Programming solution
  - LOAD dowhilepos.cpp

Activity 2

- Develop the program from practice one to..
  - ask the user to select a month from the menu, using a integer from cin
  - print the month in full for the selected value (requires IF construct)