Getting started with C++ (Part 2)
CS427: Elements of Software Engineering

Lecture 2.2

11am, 16 Jan 2012

HOW DOES COMPUTER PROGRAMMING WORK?
MAGIC.
1. Recall from last week...

2. Recall: Output

3. Recall: Variables

4. Output Manipulators
   - endl

5. Input

6. Flow of control: if-blocks

7. Loops
   - for loops

8. What is a compiler?
Recall from last week...

Please review the notes from last week. Also, please read Chapter 1 of Kelley and Pohl: *A Book on C*.

- Each program begins by including header files. In particular, we need I/O Stream:
  ```
  #include <iostream>
  ```
- Declare variables before using them. They may be local to a program block.
- The heart of the program is the `main()` function – every program needs one. `void` is the default argument list and can be omitted.
- Like C, the C++ language is case-sensitive. E.g., the functions `main()` and `Main()` are not the same.
- “Curly brackets” are used to delimit a program block.
- Every (logical) line is terminated by a semicolon;
- In strings, the backslash `escape` character is used for output. For example `\n` is used to output a new line.
- Two forward-slashes `//` indicate a comment – everything after them is ignored until an end-of-line is reached.
Recall: Output

Basic components:

- `#include <iostream>`
- `using namespace std;`
- `cout` and `cin`
- The operators `<<` and `>>`
- The identifier `cout` is an object representing the Standard Output Stream.
- The operator `<<` is the put to operator and sends the text to Standard Output – usually the terminal/console window. E.g. `cout << "Hello World." << "\n";`
Recall: Variables

**Variables** are used to temporarily store values (numerical, text, etc, ....) and refer to them by name, rather than value.

More formally, the variable’s name is called the **identifier**. It must start with a letter or an underscore, and may contain only letters, digits and underscores. **Examples:**

All variables must be defined before they can be used. That means, we need to tell the compiler before we use them. This can be done at any stage in the code, up to when the variable is first used.

Every variable should have a **type**; this tells use what sort of value will be stored in it.

The variables/data types we can define include

- Integers (positive or negative whole numbers), e.g.,
  ```
  int i; i = -1
  int j = 122;
  int k = j + i;
  ```
Recall: Variables

- Floats – these are not whole numbers. They usually have a decimal places. E.g,
  ```
  float pi=3.1415;
  ```
  Note that one can initialize (i.e., assign a value to the variable for the first time) at the time of definition.

- Characters – single alphabetic or numeric symbols, are defined using the char keyword:
  ```
  char c;    or    char s=’7’;
  ```
  Note that again we can choose to initialize the character at time of definition. Also, the character should be enclosed by single quotes.

- We can declare arrays or vectors as follows:
  ```
  int Fib[10];
  ```
  This declares a integer array called Fib. To access the first element, we refer to Fib[0], to access the second: Fib[1], and to refer to the last entry: Fib[9].

- In in C, all vectors in C++ are indexed from 0.
Here is a list of common data types. Size is measured in bytes.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>(min) Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>character</td>
<td>1</td>
</tr>
<tr>
<td>int</td>
<td>integer</td>
<td>4</td>
</tr>
<tr>
<td>float</td>
<td>floating point number</td>
<td>4</td>
</tr>
<tr>
<td>double</td>
<td>16 digit (approx) float</td>
<td>8</td>
</tr>
<tr>
<td>bool</td>
<td>true or false</td>
<td>1</td>
</tr>
</tbody>
</table>

See also: 01Variables.cpp
Recall: Variables

As with C, in C++, there is a distinction between declaration and assignment, but they can be combined. Using the const modifier means the value can’t be changed later.

Example:

Displaying the value stored in a variable is easier than in C, because we don’t have to explicitly state the data type. Example:
As well as passing variable names and strings to the output stream, we can also pass manipulators to change how variable values are displayed. Some require that `<iomanip>` be included.

- `endl` print a new line (and flush)

```cpp
int main()
{
    int i, fib[16];
    fib[0]=1; fib[1]=1;

    cout << "Without the setw manipulator" << endl;
    for (i=0; i<=12; i++)
    {
        if (i >= 2) fib[i] = fib[i-1] + fib[i-2];
        cout << "The " << i << "th " << "Fibonacci Number is " << fib[i] << endl;
    }
```
Output Manipulators

- `setw(n)` will the width of a field to `n`. Useful for tabulating data.

```
02Manipulators.cpp

```cout << "n\nWith the setw manipulator" << endl;

for (i=0; i<=12; i++)
{
    if (i >= 2) fib[i] = fib[i-1] + fib[i-2];
    cout << "The " << setw(2) << i << "th " <<
 "Fibonacci Number is " << setw(3) << fib[i] << endl;
}
```return (0);
```

Other useful manipulators:

- `setfill`
- `setprecision`
- `fixed` and `scientific`
- `dec`, `hex`, `oct`
In C we typically used `scanf` to take input from the user (via the keyboard). It’s syntax is a little difficult:

```c
scanf("%d", &i);
```

We need to use an appropriate conversion character so that the inputted data is correctly interpreted (e.g., distinguish between the number 5 and the character ‘5’).

Also, we have to remember to pass the `address` of the destination variable.

In C++ things can be a little simpler. The object `cin` is used. It is a name for the `Console Input`.

In conjunction with the operator `>>` (called the `get from` or `extraction` operator), it assigns data from input stream to the named variable.
```cpp
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

int main() {
    const double StirlingToEuro = 1.20810; // Correct 15/01/12
    double Stirling;
    cout << "Input amount in Stirling: ";
    cin >> Stirling;
    cout << "That is worth " << Stirling * StirlingToEuro << " Euros\n";
    cout << "That is worth " << fixed << setprecision(2) << "\"\u20AC" << Stirling * StirlingToEuro << endl;
    return 0;
}
```
As we will see, `cin` is an **object**, with more sophisticated uses/methods than shown here.

However, we will defer this discussion until we have studied something of **objects** and **classes**.
if statements are used to conditionally execute part of your code.

**Structure (i):**

```cpp
if( exprn )
{
    statements to execute if exprn evaluates as non-zero
}
else
{
    statements if exprn evaluates as 0
}
```
Flow of control: if-blocks

The argument to `if()` is a **logical expression**.

**Example**

- `x == 8`
- `m == '5'`
- `y <= 1`
- `y != x`
- `y > 0`

More complicated examples can be constructed using **AND &&** and **OR ||**.
Flow of control: if-blocks

```cpp
#include <iostream>

using namespace std;

int main(void)
{
    int Number;

    cout << "Please enter an integer: ";
    cin >> Number;

    if ((Number%2) == 0)
        cout << "That is an even number. " << endl;
    else
        cout << "That number is odd. " << endl;

    return (0);
}
```
Flow of control: if-blocks

More complicated examples are possible:

Structure (ii):

```c++
if( exp1 )
{
    // statements to execute if exp1 is "true"
}
el else if (exp2)
{
    // statements run if exp1 is "false" but exp2 is "true"
}
else
{
    // "catch all" statements if neither exp1 or exp2 true.
}
```
Flow of control: if-blocks

```cpp
int main(void) {
    int NumberGrade;
    char LetterGrade;

    cout << "Please enter the grade (percentage): ";
    cin >> NumberGrade;

    if ( NumberGrade >= 70 )
        LetterGrade = 'A';
    else if ( NumberGrade >= 60 )
        LetterGrade = 'B';
    else if ( NumberGrade >= 50 )
        LetterGrade = 'C';
    else if ( NumberGrade >= 40 )
        LetterGrade = 'D';
    else
        LetterGrade = 'E';

    cout << "A score of " << NumberGrade <<
         "\% cooresponds to a " << LetterGrade << "." << endl;
    return (0);
}
```
Flow of control: if-blocks

The other main flow-of-control structures are the

- `switch ... case` structures
- the use of the `?` and `:` operators.

**Exercise**

- Find out how `switch ... case` works. Rewrite the Even/Odd example above using `switch ... case`.
- What errors/bugs/problems are there with the Grades example? Use `switch` to write an improved version.
Loops

Loops in C++ are the same as in C. The most commonly used loop structure is `for`

```
for(initial value; test condition; step)
{
    // code to execute inside loop
}
```

Example: 06CountDown.cpp

```cpp
int main(void)
{
    int i;

    for (i=10; i>=1; i--)
        cout << i << " ... ";

    cout << " Zero! \n";

    return (0);
}
```
Loops

The other two common forms of loop in C++ are

- **while** loops
- **do ... while** loops

**Exercise**

Rewrite the *count down* example above using a

1. **while** loop.
2. **do ... while** loop.
What is a compiler?

...Then you need to invoke the compiler and...

What's a compiler?

...Which is then passed to a linker which turns it into machine code.

Wait... what's machine code?

Machine code is the code that can be directly executed by the CPU of the...

What's a CPU?

10 minutes later...

...So the positive charge attracts electrons from p-type silicon that separates the...

What's an electron?

30 minutes later...

...So according to this theory, electroweak symmetry breaking occurs at about 100 GeV...

What's a GeV?