CIS 190: C/C++ Programming

Introduction to C++

Outline

- Files & Compiling in C++
- Variables in C++
  - string
  - bool
- Input and Output in C++
  - cin and cout
  - file streams

Files in C++

- **`hello_world.c`**
  - becomes
- **`hello_world.cpp`**

- **`hello_world.h`**
  - stays
- **`hello_world.h`**

Compiling in C++

- instead of **`gcc`** use **`g++`**

- you can still use the same flags:

`-Wall`	for all warnings
`-c`	for denoting separate compilation
`-o`	for naming an executable
`-g`	for allowing use of a debugger

  - and any other flags you used with gcc

Outline

- Files & Compiling in C++
- Variables in C++
  - string
  - bool
- Input and Output in C++
  - cin and cout
  - file streams

Variables in C++

- leniency
  - variables can be declared anywhere
  - might still want them at the top

- new variables
  - string
  - bool

Variables in C++

- #defines still work
  - but we can use const instead

- comments can be

  `/* contained */`

  or

  `//no code on same line after`

const/#define

- **`#define`** replaces with value at compile time

```
#define PI 3.14159265358979
```

- **`const`** defines variable as unable to be changed

```
const double PI = 3.14159265358979;
```

- use in code is same for both

```
area = PI * (radius * radius);
```

Details about const

```
const double PI = 3.14159265358979;
```

- explicitly specify actual type
- a variable – so can be examined by debugger

- const should not be global
  – very very rarely

string

- requires header file: `#include <string>`

advantages over C-style strings:

- length of string is not fixed
  - or required to be dynamically allocated
- can use “normal” operations
- lots of helper functions
- not an array of characters

Creating and Initializing a string

- create and initialize as empty

```
string name0;
```

- create and initialize with character sequence

```
string name1 (“Alice”);
string name2 = “Bob”;
```

- create and initialize as copy of another string

```
string name3 (name1);
string name4 = name2;
```

“Normal” string Operations

- determine length of string

```
name1.size();
```

- determine if string is empty

```
name2.empty();
```

- can compare for equality

```
if (name1 == name2) { ... }
```

More string Comparisons

- can also use the other comparison operators:

```
if (name1 != name2) { ... }
```

- alphabetically (but uses ASCII values)

```
if (name3 < name 4) { ... }
if (name3 > name 4) { ... }
```

- and can concatenate using the '**+**' operator

```
name0 = name1 + " " + name2;
```

Looking at Sub-Strings

- can access one character like C-style strings

```
name1[0] = ‘a’;
```

- can access a sub-string

```
name1.substr(2,4);
```

  - “ice”

```
name2.substr(0,1);
```

  - “Bo”

bool

- create and initialize

```
bool boolVar1 = true;
bool boolVar2 (false);
```

- can compare (and set) to true or false

- but evaluates to 0 or 1

Outline

- Files & Compiling in C++
- Variables in C++
  - string
  - bool
- Input and Output in C++
  - cin and cout
  - file streams

Working with Input/Output in C++

- at top of each file that uses input/output

  `using namespace std;`

- to use streams to interact with user/console, must have `#include <iostream>`

- to use streams to interact with files, must have `#include <fstream>`

Input/Output in C

- `#include <stdio.h>`

- `printf(“test: %d\n”, x);`

- `scanf(“%d”, &x);`

Streams in C++

- ~~`#include <stdio.h>`~~
  - `#include <iostream>`

- ~~`printf("test: %d\n", x);`~~
  - `cout << "test: " << x << endl;`

- ~~`scanf("%d", &x);`~~
  - `cin >> x;`

More about C++ Streams

- in order to use C++ streams as shown
  - at top of each file you must have

    `using namespace std;`
  - otherwise you must use

    `std::cin, std::cout, std::endl`

- in addition to **`cin`** and **`cout`**, we have **`cerr`**
  - instead of **`fprintf(stderr, “error!”);`**

Reading In Files in C

- `FILE *ifp;`

- `ifp = fopen("testFile.txt", "r");`

- `if ( ifp == NULL ) { /* exit */ }`

- read specified in call to **`fopen()`**

Reading In Files in C++

- ~~`FILE *ifp;`~~
  - `ifstream inStream;`
- ~~`ifp = fopen(“testFile.txt”, “r”);`~~
  - `inStream.open(“testFile.txt”);`
- ~~`if ( ifp == NULL ) { /* exit */ }`~~
  - `if (!inStream) { /* exit */ }`

- read specified by variable type
  - `ifstream` for reading

Writing To Files in C

- `FILE *ofp;`
- `ofp = fopen("testFile.txt", "w");`
- `if ( ofp == NULL ) { /* exit */ }`

- write specified in call to **`fopen()`**

Writing To Files in C++

- ~~`FILE *ofp;`~~
  - `ofstream outStream;`
- ~~`ofp = fopen("testFile.txt", "w");`~~
  - `inStream.open("testFile.txt");`
- ~~`if ( ofp == NULL ) { /* exit */ }`~~
  - `if (!outStream) { /* exit */ }`

- write specified by variable type
  - `ofstream` for writing

Using Streams in C++

- must have **`#include <fstream>`**
- once file is correctly opened, use **`inStream`** and **`outStream`** the same as **`cin`** and **`cout`**

```
inStream >> firstName >> lastName;
outStream << firstName << " "
          << lastName << endl;
```

Advantages of Streams

- does not use placeholders (**`%d`**, **`%s`**, etc.)
  - no placeholder type-matching errors
- can split onto multiple lines
- precision with printing can be easier
  - once set using **`setf()`**, the effect remains until changed with another call to **`setf()`**

Finding EOF with ifstream – Way 1

- use **`cin`**'s boolean return to your advantage

```
while (inStream >> x)
{
   // do stuff with x
}
```

Finding EOF with ifstream – Way 2

- use a “priming read”

```
inStream >> x;

while( !inStream.eof() )
{
   // do stuff with x

   // read in next x
   inStream >> x;
}
```

The `>>` Operator

- returns a boolean for (un)successful read

- just like scanf and fscanf:
  - skips leading whitespace
  - stops at the next whitespace (without reading it in)

hello_world.cpp

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

int main()  {
  cout << “Hello world!”
       << endl;

  return 0;
}
```

Next Few Classes

- vectors
- header protection
- classes
- operator overloading
- new/delete
- and more!

Homework 4B

- due this coming Wednesday
- any questions?