Kevin Chen

10/30/15

C++ Notes (Abbreviated)

Basics:

* #include
  + Lines beginning with a hash sign are directives read and interpreted by the preprocessor. #include tells it to include a section of standard C++ code.
* using can help abbreviate code.
  + std::cout << “Hello world” – OR –
  + using namespace std;

cout << “Hello world”

* cout
  + prints to the console
  + Example:

Cout << “Hello world! ” << endl;

* + endl **end**s the **l**ine (printing a newline character and flushing the stream)

Variables:

* Fundamental types:

|  |  |  |
| --- | --- | --- |
| **Group** | **Type names** | **Notes on size / precision** |
| Character types | **char** | 8 bits. |
| **char16\_t** | 16 bits. |
| **char32\_t** | 32 bits. |
| **wchar\_t** | Can represent the largest supported character set. |
| Integer types (signed) | **signed char** | 8 bits. |
| *signed* **short** *int* | 16 bits. |
| *signed* **int** | 16 bits. |
| *signed* **long** *int* | 32 bits. |
| *signed* **long long** *int* | 64 bits. |
| Integer types (unsigned) | **unsigned char \*** | (same size as their signed counterparts) |
| **unsigned short** *int* |
| **unsigned** *int* |
| **unsigned long** *int* |
| **unsigned long long** *int* |
| Floating-point types | **float** |  |
| **double** | More precise than float |
| **long double** | More precise than double |
| Boolean type | **bool** |  |
| Void type | **void** | no storage |
| Null pointer | **decltype(nullptr)** |  |

\* the only type that is guaranteed to be one byte.

* Declaration
  + Can declare int a, b, c; // Creates three int variables.
* Initialization – all three of these mean the same thing:
  + int a = 5;
  + int a(5);
  + int a{5};
* Type deduction: auto and decltype – don’t actually have to specify type
  + auto:

int foo = 0;

auto bar = foo; // the same as int bar = foo;

* + decltype:

int foo = 0;

decltype(foo) bar; // the same as int bar;

* Strings
  + #include <string>
  + A compound type: basically a struct/object.

Constants:

Literals

* express particular values within the source code of a program
* Example: a = 5 // The “5” is the literal.
* **Integer Numerals:**
  + Bases:
    - 75 // Decimal
    - 0113 // Octal – begins with “0”
    - 0x4b // hexadecimal – begins with “0x”
  + Suffix types:

|  |  |
| --- | --- |
| **Suffix** | **Type modifier** |
| u *or* U | unsigned |
| l *or* L | long |
| ll *or* LL | long long |

* + - 75u // unsigned int
    - 75l // long
* **Floating Point Numerals:**
  + Powers:
    - 3.14159 // 3.14159
    - 6.02e23 // 6.02 x 10^23
  + Suffix types:

|  |  |
| --- | --- |
| **Suffix** | **Type** |
| f *or* F | float |
| l *or* L | long double |

* + - 3.14159L // long double
    - 6.02e23f // float
* **Character and string:**
  + Single quotes for char, double quotes for strings
  + Prefix types:

|  |  |
| --- | --- |
| **Prefix** | **Character type** |
| u | char16\_t |
| U | char32\_t |
| L | wchar\_t |

* + Other prefixes:

|  |  |
| --- | --- |
| **Prefix** | **Description** |
| u8 | The string literal is encoded in the executable using UTF-8 |
| R | The string literal is a raw string |

* + Raw string: backlashes, single, double quotes are all valid.
    - R“[sequence]([content])[same sequence]”
* **Other types**
  + bool: *true* and *false*
  + pointer: *nullptr*

Preprocessor Definitions (#define)

* #define [identifier] [replacement value]
* Example:

#define PI 3.1415

double a = PI;

**Operators**

* Assignment Operator (=) returns the value on the right.
  + Example: y = 2 + (x = 2) sets int x to 2 and int y to 4, and returns 4.
* Increment and Decrement
  + ++x vs. x++: ++x increments x and returns the new value of x. x++ also increments x but instead returns the original value of x.
* Conditional ternary operator
  + Format: condition ? result1 : result2
  + Returns result1 if condition is true, returns result2 is false.
  + Example: 7 == 5 ? 2 : 3 returns 3. 7 == 7 ? 2: 3 returns 2
* Comma Operator
  + Used to separate two or more expressions when only one is expected.
  + All expressions except the right one are first evaluated. Then the rightmost expression is returned.
  + Example:

a = (b=3, b+2); b = 3, then return b + 2, which is 5. So a = 5, b = 3

|  |  |  |
| --- | --- | --- |
| **operator** | **asm equivalent** | **description** |
| & | AND | Bitwise AND |
| | | OR | Bitwise inclusive OR |
| ^ | XOR | Bitwise exclusive OR |
| ~ | NOT | Unary complement (bit inversion) |
| << | SHL | Shift bits left |
| >> | SHR | Shift bits right |

* Explicit Casting operators
  + int i = (int) 3.14; – OR – int i = int (3.14);
* sizeof operator
  + Returns size of type/variable/object in bytes.
  + Example: x = sizeof(char) returns 1, since char is one byte.
* Operator Precedence can be found online

**Basic Input/Output**

|  |  |
| --- | --- |
| **stream** | **description** |
| cin | standard input stream |
| cout | standard output stream |
| cerr | standard error (output) stream |
| clog | standard logging (output) stream |

* cout: cout << “hello ” << “world!”; // prints out “hello world!”
  + Used with insertion operator: <<
  + Does not automatically make line breaks at the end, unless instructed to do so.
  + Can use the new line character “\n” or “endl” manipulator to create new line.
  + “endl” flushes out the stream’s buffer (if any), while \n does not.
* cin: cin >> age // stores user input into “age”.
  + Can be chained for multiple requests.

Example: cin >> a >> b; is equivalent to cin >> a; cin >> b;

* + If you give an invalid input (e.g. string into an integer variable), then the program continues without doing anything.
  + Spaces terminate the value being extracted. To get an entire line, use: getline(cin, [variable name]);
* stringstream: allows us to treat strings as streams, and thus convert between strings and numerical values.
  + Example:

getline(cin, mystr);

stringstream(mystr) >> price;