Southern New Hampshire University

CS 260 C++ Primer

The C programming language was developed in the early 1970s as a small, powerful, and portable[1] programming language with relatively few keywords that mapped directly onto the underlying machine instructions. At that time, achieving small and fast execution meant writing in assembly language—a low-level programming language supporting instructions of a specific processor. Thus, assembly language programs were not portable between different processor architectures, instead requiring a program to be completely rewritten for another system. The UNIX operating system was written entirely in C, becoming the first truly portable operating system, able to run on different processor architectures from different computer hardware vendors.

C was revolutionary for its simplicity and portability, but it was also notoriously difficult to create well-written, error-free code because of its simplicity and weak type system. C is a procedural language, meaning the programmer completely controlled program flow using procedures, or subroutines (functions), to group logic together. Procedural programs often lead to unreadable and unmaintainable code (referred to as spaghetti code[2]) due to the wide variety of coding styles and idiosyncrasies of individual programmers.

While computer scientists developed the ideas of objects and object-based programming methodologies in the 1960s and 1970s, it was not until the mid-1980s that Bjarne Stroustrup released “C with Classes,” later renamed to “C++,” where “++” was a familiar increment operator from the C language. C++ compilers were mostly backwards-compatible, making it easier for programmers to begin using the new tools while slowly adopting the new language features, including:

- Constructors/Destructors
- Function & Operator Overloading
- Standard Library
- Exception Handling
- Name scoping
- Templates

In contrast, Java was released in the mid-1990s as a fully object-oriented language that borrowed the syntax of C and C++ without the difficult portions (low-level memory management, pointers, multiple inheritance) while adding concurrency, documentation generation, a significantly larger standard library, security, simplicity (e.g., data types) and the portability of byte-code class files due to the Java Virtual Machine (JVM) layer. Later, C# was announced in 2000 by Microsoft as their premier object-oriented language competing with Java and C++.

[1] Portability is achieved at the “source code level” by having C compilers available for nearly all processors and operating systems. The source code may be loaded, compiled, and executed on a different operating system than it was originally written and tested on.

[2] The seminal work on procedural programming is Edsger Dijkstra's letter to the editor of ACM, Go To Statement Considered Harmful.

Basic data types:

Data Types	C++	Java	Size (bits)	Range / Notes
Boolean, two states	bool	boolean	1 bit	0/1
Single character	char	char	2 bytes (16)	
Wide character[1]	wchar_t		4 bytes (32) Linux 2 bytes (16) Windows	
String	char[] std:string	 String	varies	C-style null-terminated character array C++ string defined using new template feature[2]
Integer	int	int	4 bytes (32)	
Floating point	float	float	4 bytes (32)	$1.40239846 \times 10^{-45}$ to $3.40282347 \times 10^{38}$
Double-wide floating point	double	double	8 bytes (64)	$4.9406564584124654 \times 10^{-324}$ to $1.7976931348623157 \times 10^{308}$
No value	void	void	n/a	

[1] Most modern languages and operating systems support Unicode today; however, in C++, it is still very tricky and troublesome.

[2] C-style string can be accessed using ".c_str()" function of a std::string; for example, firstName.c_str().

C++ type modifiers:

Modifier	C++	Java	Size	Range
signed	signed int		4 bytes (32) [1]	-2^{31} to 2^{31}-1
unsigned	unsigned int		2 bytes	
	signed char	byte	1 byte	-128 to 127
	unsigned char		1 byte	0 to 255
short	short int	short	2 bytes	-32,768 to 32,767
	unsigned short int		2 bytes	0 to 65,535
	signed short int		2 bytes	-32,768 to 32,767
long	long int	int	8 bytes	-2,147,483,648 to 2,147,483,647
	signed long int	long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
	unsigned long int		8 bytes	0 to 18,446,744,073,709,551,615
constant	cont	final		
enumeration		enum		

[1] One bit of a byte is reserved to hold the sign.

Conditionals:

	C++	Java
if	`if (condition) { // do something; }`	`if (condition) { // do something; }`
if / else	`if (condition) { // do something; } else { // do something else; }`	`if (condition) { // do something; } else { // do something else; }`

Loops:

Loop	C++	Java
for	`for (initial, test, increment) { // do something; }`	`for (initial, test, increment) { // do something; }`
while	`while (condition) { // maybe do something; }`	`while (condition) { // maybe do something; }`
do while	`do { // do something at least once; } while (condition)`	`do { // do something at least once; } while (condition)`

Input/Output:

C++	Java
`std::cin >> firstName;` *or* `std::getline(std::cin, firstName);`	`Scanner scnr = new Scanner(System.in);` `firstName = scnr.next();`
`std::cout << firstName;`	`System.out.print(`*firstName*`);`
`std::cout << firstName << std::endl;`	`System.out.println(`*firstName*`);`

Objects:

	C++	Java
Declare and initialize using the default constructor	`Foo foo;`	`Foo foo = new Foo();`
Declare and initialize using an overloaded constructor	`Foo foo(value);`	`Foo foo = new Foo(value);`
Modify an object's member	`foo.firstName = “John”;`	`foo.firstName = “John”;`
Reference another instance of the same object type (bar will also point to the foo object instance)	`Foo &bar = *foo;`	`Foo bar = foo;`
Copy contents of an object instance into a new object instance	`Foo bar = foo;`	`Foo bar = foo.clone();`
Declare a reference to an object	`Foo *foo;` `// undefined until pointer is set`	`Foo foo;` `// foo is null until initialized`
Refer to a member of a reference pointed object	`foo->firstName;`	`foo.firstName`

Create a Type Definition:
A type definition is like an “alias name” for a type. This technique lets you create shorter, friendlier names scoped locally:

```
// create a friendly name for the std::string class
typedef std::string String;

// use the typedef to declare a variable
String firstName;
```

Note: For anything beyond trivial programs, this is the preferred approach to using shorthand. Importing the namespace means the entire namespace becomes imported, which may conflict with other code/libraries/routines being used by your program.

Display Contents of an Array:

```
for (int i = 0; i < numElements; i++) {
       cout << array[i] << endl;
}
```

Display Contents of a Vector:

```
vector<int> primes = {2, 3, 5, 7, 11, 13, 17, 19, 23};
for (auto const& c: primes) {
       cout << c << endl;
}
```

Note: This only works with simple types like int or string; complex types (struct, class) need individual fields specified separately.

Measuring Elapsed Time:
When considering algorithms and different approaches to solving problems, it may be helpful to measure the elapsed time that each approach takes. A simple technique to do so involves three steps:

1. Import the time library:
   ```
   #include <time.h>
   ```
2. Declare an instance of the *clock tick* type (units of time of a system-specific length) and initialize the starting time before (above) the start of the code to be measured:
   ```
   clock_t ticks;
   ticks = clock();
   ```
3. Calculate the ending time after (below) the code to be measured has completed, and display the results:
   ```
   ticks = clock() - ticks; // current clock ticks minus starting clock ticks
   cout << "time: " << ticks << " milliseconds" << endl;
   cout << "time: " << ticks*1.0/CLOCKS_PER_SEC << " seconds" << endl;
   ```

Parsing Comma-Separated Value (CSV) Files:
There are several existing libraries and routines for parsing CSV files, as well as creating the code from scratch, which is not as easy as you might think. One of the simpler libraries is CSVParser, written by Romain Sylvain and released under the MIT License. It handles the drudgery of opening the file, reading the data rows, parsing the values for each row, and presenting them to the calling program as a vector of values easily accessed using syntax such as file[1][4] for second row, fifth column.

Pass Arguments on the Command Line:
A common technique for testing a C++ class is to put test code into the main() method and then run the class from a command-line console. It may be helpful to pass arguments from the command line to alter the testing behavior—for example, passing in the path to an input file containing data to use when testing. We will employ this technique by starting with a small CSV file containing fewer than 200 bids and later scaling up to testing with nearly 10,000 bids.

```
int main(int argc, char* argv[]) {
	string csvPath, bidKey;

	switch (argc) {
	case 2:
		csvPath = argv[1];
		bidKey = "98109";
		break;
```

```
        case 3:
                csvPath = argv[1];
                bidKey = argv[2];
                break;
        default:
                csvPath = "eBid_Monthly_Sales_Dec_2016.csv";
                bidKey = "98109";
        }
```

Converting a String Value to a Double:

```
double strToDouble(string str, char ch) {
        str.erase(remove(str.begin(), str.end(), ch), str.end());
        return atof(str.c_str());
}
```

Usage: `amount = strToDouble(money, '$');`

Note: The above function uses C++ 11 features, so remember to add the compiler option as described in the C++ Development Installation document.

References

C++. (2017, February 3). In *Wikipedia, The Free Encyclopedia*. Retrieved 19:59, February 4, 2017, from https://en.wikipedia.org/w/index.php?title=C++&oldid=763408990

Kaltofen, E. (2013, October 29). Erich's Java cheat sheet for C++ programmers. Retrieved February 04, 2017, from http://www4.ncsu.edu/~kaltofen/courses/Languages/JavaExamples/cpp_vs_java/

McCown, F. (n.d.). Java and C# Comparison. Retrieved February 04, 2017, from http://www.harding.edu/fmccown/java_csharp_comparison.html

Radeck, K. (2003, October). C# and Java: Comparing Programming Languages. Retrieved February 04, 2017, from https://msdn.microsoft.com/en-us/library/ms836794.aspx

Various. (2006, August 9). C++ Quick Reference Sheet (Cheat Sheet) – C++ Tutorials | Dream In Code. Retrieved February 04, 2017, from http://www.dreamincode.net/forums/topic/12694-c-quick-reference-sheet-cheat-sheet/