Overview of C, C++ and Java

Programming Language Design and Implementation

(4th Edition)

by T. Pratt and M. Zelkowitz

Prentice Hall, 2001

Book sections:

- Section 1.5
- Section 2.2.4
- Section 6.5
- Appendix A.2
- Appendix A.3
- Appendix A.5

Three generations of programming language

These three languages all have the same basic syntax. They were developed at different times and the differences between them reflect the changes in what was thought to be important in a programming language.

Straight from the Horses Mouth

The C Programming Language by Brian W.Kernighan and Dennis M. Ritchie, Prentice-Hall, 1988.

The C++ Programming Language by Bjarne Stroustrup, Addison Wesley, 1997.

Sun's Java website at `http://java.sun.com/`

C overview

- C developed in 1972 by Dennis Ritchie and Ken Thompson at AT&T Bell Telephone Laboratories
- Developed as language to implement UNIX on a DEC PDP-11. UNIX was a then "small" operating system to compete with the large Multics of MIT and GE.

C is more of an environment than a simple language

- The C language
- The C preprocessor (#include, #if, ...)
- The C interface assumptions (.h include files)
- The C library ("Built-in" functions like printf, malloc, ...)

C program structure

- C program is a sequence of procedures and global declarations
- Each procedure contains
  - local declarations
  - imperative statements, which can call other procedures
- Most data are integer data. This allows full flexibility since almost everything is an integer.

Reserved Words common to C and C++

auto	break	case	char	const	continue
default	do	double	else	enum	extern
float	for	goto	if	return	short
signed	sizeof	static	struct	switch	typedef
union	unsigned	void	volatile	while	

C program structure

```
#include <stdio.h>
#include "myHeader.h"

int main( int argc, char argv[][])
{
/* body of program goes here
   a sequence of declarations ans statements
*/

return 0;
}

/* other function definitions */
```

Compiling and running a C program

The C compiler we use under Linux is the Gnu C compiler which is called `gcc`.

To compile a program contained in a single file called `prog.c`, type

```
gcc proc.c
```

The default name for the executable file is `a.out`. To give the output file a different name, type

```
gcc -o prog proc.c
```

To run the executable file, just type its name

```
prog
```

or possibly

```
./prog
```

Today C and to some extent C++ have become the dominant language used to develop systems software (operating systems, compilers, utility programs, video games, etc.)

C++ overview

Developed by Bjarne Stroustrup at AT&T in 1986 as an extension to C.

- C++ includes essentially all of C (both good and bad features)
  - Improves on C by adding strong typing
  - Goal was to keep efficiency of C execution, while adding new features
- C++ adds concept of a class (borrowed from Simula, which was a simulation language developed from Algol).
  - Data was defined local to a class
  - Functions (methods) could access this local data
  - Implemented concept of inheritance of classes

C/C++ Compatibility

Although designed to be compatible originally, the two languages are drifting apart with time.

For a discussion of the issues, see recent articles by Bjarne Stroustrup in the C/C++ Users Journal

- C and C++: Siblings (July 2002, p 28.)
- C and C++: A Case for Compatibility (August 2002, p 22.)
- C and C++: Case Studies in Compatibility (September 2002, p 22.)

C++ also more of an environment than a simple language

- The C++ language
- The C++ preprocessor (#include, #if, ...)
- The C++ library ("Built-in" functions and classes like iostream and string
- Standard Template Library (STL) with a collection of container classes and utilities

C++ Reserved Words

asm	bool	catch	class	const_cast	delete
dynamic_cast	explicit	false	friend	inline	mutable
namespace	new	operator	private	protected	public
reinterp_cast	static_cast	template	this	throw	true
try	typeid	typename	using	virtual	wchat_t

C++ program structure

The same as for a C program but include different header files

```
#include <iostream.h>
#include "myHeader.h"

int main( int argc, char argv[][])
{
/* body of program goes here
   a sequence of declarations ans statements
*/

return 0;       // signals successful execution to Linux
}

// other function definitions
```

Compiling and running a C++ program

The C compiler we use under Linux is the Gnu C compiler which is called `g++`.

To compile a program contained in a single file called `prog.c`, type

```
g++ proc.c
```

The default name for the executable file is `a.out`. To give the output file a different name, type

```
g++ -o prog proc.c
```

To run the executable file, just type its name

```
prog
```

or possibly

```
./prog
```

Java development

Java development began in 1991 at Sun Microsystems, where James Gosling led the Green Team in a project to develop a language for use on consumer digital devices.

In 1993, the Mosaic Web Browser was released. The Green Team immediately saw the role of their language as a way to enhance Web browsers.

Because of slow transmission speeds, Sun saw this new language as a way to incorporate executable behavior in the browser rather than in the web server.

A critical development: The Web server would not know what machine the user's browser was executing on. To solve this problem, a Java virtual machine was designed and the applet would be compiled into a series of bytecodes for that virtual machine.

By 1995, Java became a de facto standard for web browser applets.

Java Overview

For the most part, Java is C++ with the excess baggage of C removed.

- Data are strongly typed, with integers, Booleans, and characters all being separate types.
- Arrays are separate types, and a string is not simply an array of characters.
- Method invocation is the only subroutine linkage. Because all objects are part of classes, there is no need for separate function or procedure calls.
- Struct objects are not needed because the same effect can be achieved via instance variables in class definitions.
- Pointers are implicit in the language, but there is no pointer data type.

Java program structure

A Java application consists of one or more classes. Only import statements exist outside of a class.

Each class has the structure shown below.

Generally, each public class is put into a separate file.

The name of the file and the name of the class have to match. That is the class MyClass needs to be in a file called `MyClass.java`.

Java Reserved Words

boolean	break	byte	case	catch	char
class	const (not used)	continue	default	do	double
else	extends	false	final	finally	float
for	goto (not used)	if	implements	import	instanceof
int	interface	long	native	new	null
package	private	protected	public	return	short
static	strictfp	super	switch	synchronized	this
throw	throws	transient	true	try	void
volatile	while				

```
import java.io.*;

/** class comment for javadoc */
public class myClass
{
    // declarations for static and instance variables

    /** method comment for javadoc
              every application needs a main method */
    public static void main( String [] args)
    {
    /* body of main method goes here
       a sequence of declarations and statements */
    }

    // other method definitions
}
```

Compiling and running a Java program

Java programs are compiled to bytecode using the `javac` program.

```
javac MyClass.java
```

To a limited extent, javac will find and compile the other classes needed by the class being compiled. For complex applications, you may need to force everything to be compiled by typing

```
javac *.java
```

javac generates a class file, MyClass.class in the above example. To run the application, type

```
java MyClass
```

where MyClass is the class ciontaining the main method.

There is a utility called `javadoc` which will automatically generate documentation in HTML format using the special javadoc comments. The following puts the documentation into a directory called html which must already exist.

```
javadoc -author -version -d html/ *.java
```

Scalar data in C and C++

- Numeric data can be signed or unsigned and several different sizes are provided: short, int, long int, long long int, unsigned int, unsigned long.
- Floating point data: float, double, long double.
- Character data are stored as type char using the ASCII code.

  ```
  char ch = 'a';
  ```

  creates a variable ch with the ASCII value for the character a stored in the memory location associated with ch.
- Boolean data is stored in int format; 0 is false and everything else is true. C++ provides a bool type and the keywords true and false to represent the two possible values.

Java Scalar data

- Numeric data can be 8-bit, 16-bit, 32-bit or 64-bit signed integer data. There are also 32-bit float and 64-bit double real data that follow IEEE Standard 754.
- Java also stores data as byte and short, although these are coerced into int to perform any operations on them.
- Character data are stored using unicode. Unicode is an international standard that creates 16-bit characters. For example,

  ```
  char x = 'a'
  ```

  creates an object x with the unicode value for the character a.
- Boolean data may be true or false. Unlike C, Boolean is not an int.

Control structures

The control structures (selection and repetition) have the same basic syntax in all three languages.

- Selection
  - if and if-else

```
if (condition)
    ifBody;
else     /* else part is optional */
    elseBody;
```

  - switch –control variable must be of enumerable type

```
switch (data) {
    /* one case for each useful data value
    case value1: action1; break;
    case value1: action1; break;
    default: defaultAction;
    }
```

- Repetition
  - while

```
while (condition)    /* condition tested first */
    whileBody;
```

  - do-while

```
do    /* body executed first */
    doBody;
while (condition);    /* condition tested last */
```

  - for – compact form of while loop

```
for (init; condition; update)
    forBody;
```

Input and Output

C uses a FILE type for sequential files.

C++ and Java both use streams for input and output.

C standard I/O

include file stdio.h

Standard input file (keyboard) is stdin

formatted read uses `scanf`

Standard output file (console) is stdout

formatted output uses `printf`

printf and scanf use format strings to specify the format. For each data type, there is a format specifier which is one or two characters preceded by a `%`

Type	output specifier	input specifier
int	%d	%d
double	%f	%lf
char	%c	%c
string	%s	%s

C++ standard I/O

include file iostream.h

Standard input stream (keyboard) is cin

Extraction operator >> takes data from stream and stores it in a variable.

Standard output stream (console) is cout

Insertion operator << takes data from a variable and appends it to the stream.

Java standard I/O

Standard input stream (keyboard) is System.in – a byte-oriented stream which needs to be transformed into a BufferedReader (java.io package) for easy use.

```
    BufferedReader in = new BufferedReader(
new InputStreamReader( System.in));
```

Use the readLine method to get a line of input:

```
    String inputLine = in.readLine();
```

Standard output stream (console) is System.out which can be used directly. Do output using `println` method:

```
    System.out.println(
"This string will be output to the console");
```

formatting requires a special format object – DecimalFormat for general floating point formats, NumberFormat allows for percentage and currency formats

Files in C

include fileio.h

Declare files with

```
FILE * fin;
FILE * fout;
```

Connect to physical files with

```
fin = fopen( "datafile" "r");
fout = fopen( "outfile" "w");
```

Any problems opening the file result in the FILE variable being NULL.

Reading and writing are done with `fscanf` and `fprintf` which are similar to the functions used for standard I/O but with a FILE * variable as the last argument.

Files can be closed with fclose.

```
fclose( fin);
```

Files in C++

include fstream.h The filestream classes inherit from the istream and ostream classes used for standard I/O.

Open a file with the constructor or the open function.

```
ifstream fin( "datafile");

ofstream fout;
fout.open( "outfile");
```

Use the insertion and extraction operators as before but replace cin and cout by the filestream object names.

Close a file with the close function.

```
fout.close();
```

Files in Java

Use a FileReader object which is connected to the physical file by the constructor or the open method.

```
FileReader fileReader = new FileReader( args[0]);
```

Create a BufferedReader from the FileReader and use the readLine method

```
BufferedReader infile = new BufferedReader( fileReader);
```

Use a FileWriter object which is connected to the physical file by the constructor or the open method.

```
FileWriter fileWriter = new FileWriter( args[1]);
```

Create a PrintWriter from the FileWriter and use the readLine method

```
PrintWriter outfile = new PrintWriter( fileWriter);
```

Close a file with the close method.

```
infile.close();
```

Strings in C

C uses arrays of characters to represent strings. The null character, `'\\0'` which has ASCII code 0, is used to terminate a string.

include string.h

The null character will be appended automatically by scanf, during assignment and by the functions in string.h

You really need to understand pointers to use the C string functions.

The programmer is responsible for checking that there is enough space in the destination string argument of the functions in string.h

Strings Functions in C

`int strlen( char str[])` : returns the number of characters in the string, not including the null character

`void strcpy( char destination[], char source[])` : copy the source string into the destination

`void strncpy( char destination[], char source[], size_t n)` : copy up to n characters of the source string into the destination

`void strcat( char destination[], char source[])` : append the source string into the destination

`char * strrchr( const char str[], const char ch)` : points to the first occurrence of ch in str. If you need an index to that postiion, the returned valur - the original string name will give it to you.

`char * strtok( char str[], const char str2[])` : a function that can be used to break a string up into tokens using the characters in str2 as delimiters. This destroys the original string by writing over the first delimiter found. It returns a pointer to the remainder of the string.

Strings in C++

C++ has a string class with methods to do most of what you need to do.

```
include <string>
using namespace std;
```

Note that there is no .h

strings can be declared and created in several ways:

```
string str( "a string");
string str2 = "another string";
```

You can use assignment (=) and concatenation (+) with C++ strings.

Class functions for the C++ string class

`int size()` returns the number of characters in the string

`char at( int)` returns the character at the given position

`char* c_str()` returns a C-style string representation of the string; you need to copy it into another string before you can do much of anything with it.

`int find( char)` `int find( string)` `int find (char[])` allows you to find the position of a particular char, string or char[] in the array

`string substr( int start, int length)` returns a length-character substring starting at position start

`void insert( int, char[])` insert the string parameter into the string starting at the position given

`void replace( int pos, int n, char[])` replace the n characters beginning at position pos with the given string

`void erase( int pos, int n)` remove n characters starting at position pos

Strings in Java

Java also has a String class which is in the java.lang package. You do not need to import anything.

Create a String in two ways

1. Using a constructor

```
String str1 = new String( "some text"
```

2. Assignment to a literal string

```
String str2 = "some other text"
```

String methods in Java

`char charAt( int n)` : returns character n (counting from 0) of the String

`int length()` : returns the number of characters in the String

`int compareTo( String)` : returns -1, 0 or 1 depending on the lexicographic ordering of the two strings

`String concat( String)` : returns a new String that is the concatenation of the original with the parameter

`boolean equals( String)` : case-sensitive check for equality

`String substring( int offset, int endIndex)` : returns a substring whose last character was at index-1

String I/O

C and C++ read whitespace delimited strings by default. To get a single character at a time, use a char variable.

To get an entire line at a time, use special functions

char * fgets(char *, nChars, inputFILE)

A BufferedReader in Java reads one line at a time. To break it up into smaller pieces, use a StringTokenizer. The default delimiter is whitespace but you can specify whatever you want.

Another way is to use a StreamTokenizer.

We'll continue from here later

Arrays in Java

- Arrays are dynamic and are allocated as pointers:
- Declare an empty array:

```
Newvalues Mydata[ ]
```

- To allocate the array: Call the new function to allocate a 100-element array.

```
Mydata = new Newvalues[100];
```

- Both could be accomplished in one step; for example,

```
int[ ] Mydata = new int[100];
```

- To actually allocate each element of the array, you would also need to do, for each i,

```
Mydata[i] = new Newvalues()
```

Classes

Class definitions similar to C++ class definitions in defining instance variables and methods for a new class, but like Smalltalk the superclass of a new class is explicitly given:

```
class Newvalues extends Object {
    public double  x; /*  instance variable  */
    public int  y; /*  instance variable  */

    Newvalues() { . . .  /* Constructor for Newvalues */}
}
```

Predefined classes

- Java uses a large predefined class library.
  -
  - java.io.* is the standard input-output library
  - java.awt.* is the standard applet library
- Java has the same sub-classing structure of C++. A subclass Newclass is a subclass of Olderclass by the class definition
  - class Newclass extends Olderclass . . .
- newclass inherits all the instance variables of Olderclass
- method invocation to Newclass (as in Newclass.functionX) is passed to Olderclass.functionX if functionX is not a method defined in class Newclass.

Instance variables

Variables in a class definition are usually instance variables- they are in every instance of an object of that class.

Static attribute - To have a variable that is common across all instances of a given class. For example, a linked list structure would be of class LinkedObject. static FreeListType FreeList;

Java virtual methods are called abstract and are defined similarly to C++ virtual methods: abstract void MethodName(); /* Null body */

Threads

- Java allows for some level of multiprocessing through threads.
- Threads are defined by the

  `new Thread(object) command.`
- The synchronized attribute

  `(synchronized void startSort())`

  on multiple-method definitions prevents more than one of them from executing at a time, and thus avoids deadlock situations.

Threads

- Created by the new command
- Executed by the run() method; started with start()
- Executes until the stop() method or run() completes execution
- Threads may suspend() and resume() execution.
- Synchronization: Use monitors with synchronized attribute:

```
public synchronized void methodName(int value)
{val= returned_value;
```