

Baby Steps: Some Simple Programs in C, C++, and Java

Two Goals:

- To introduce the rudiments of OO vocabulary by showing the easily established conceptual parallels between C, C++, and Java programs.
- To familiarize you with command-line compilation of C++ and Java programs.

Simple Programs: Summing an Array of Integers

```
/* AddArray1.c */

#include <stdio.h>

int addArray( int [], int );

main()
{
    int data[] = {4,3,2,1,0,5,6,7,8,9};
    int size = sizeof(data)/sizeof(data[0]);
    printf("sum is %d\n", addArray( data, size ));
    return 0;
}

int addArray( int a[], int n ) {
    int sum = 0;
    int i;
    for(i=0; i<n; i++ )
        sum += a[i];
    return sum;
}
```

```
/* AddArray2.c */

#include <stdio.h>

int addArray( int*, int );

main()
{
    int data[] = {4,3,2,1,0,5,6,7,8,9};
    int size = sizeof(data)/sizeof(data[0]);
    printf("Pointer: sum is %d\n", addArray( data, size ));
}

int addArray( int* a, int n ) {
    int sum = 0;
    int i;
    for(i=0; i<n; i++ )
        sum += *(a + i);
    return sum;
}
```

```
//AddArray.cc
```

(A)

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

int addArray( int*, int );

int main()
{
    int data[] = {4,3,2,1,0,5,6,7,8,9};
    int size = sizeof(data)/sizeof(data[0]);
    cout << "C++ version: sum is "
         << addArray( data, size ) << endl;
}
```

```
int addArray( int* a, int n) {
    int sum = 0;
    int i;
    for(i=0; i<n; i++ )
        sum += *(a + i);
    return sum;
}
```

For C++ compilation:

The filename must end in .cc or .C or .cpp

Recommended compiler for C++:

g++ (The GNU C++ compiler)

For Unix/Linux, you can download the latest version from www.gnu.org

For Windows OS, you can download it from

<http://sourceware.cygwin.com/cygwin>

This download will include many unix emulation tools for Windows machines.

On Solaris, you can also use the CC compiler, but it is very outdated.

A style difference between C and C++:

The **main** in both C and C++ returns the status code, which is 0 for normal program termination and some non-zero integer value for abnormal termination.

By tradition, the return value of **main** in C programs is left unmentioned – it being **int** by default.

On the other hand, the convention in C++ is to explicitly mention the return type of **main**.

```
//AddArray.java
```

```
public class AddArray { // (A)

    public static void main( String[] args ) // (B)
    {
        int[] data = { 0, 1, 2, 3, 4, 5, 9, 8, 7, 6 }; // (C)
        System.out.println( "The sum is: " // (D)
                            + addArray(data) );
    }

    public static int addArray( int[] a ) { // (E)
        int sum = 0;
        for ( int i=0; i < a.length; i++ )
            sum += a[i];
        return sum;
    }
}
```

In Java, the name of the file will usually be same as the name of the class, except for the suffix which must be `.java`.

This is only necessary for a class that is declared to be public. A file is only allowed to hold one public class.

But if a file does not have a public class, you can call the file anything (provided it carries the suffix `.java`).

If you'll be doing your homework on your home machine, download the latest Java compiler from

<http://java.sun.com/products/products.a-z.html>

and download

Java 2 SDK, Standard Edition, v. 1.4

You invoke the compiler by

```
javac filename.java
```

The compiler outputs what is known as the bytecode for the classes in your `.java` file.

For our example, the output will be

```
AddArray.class
```

Unlike the executables for C and C++, the bytecode is machine independent. It is run by invoking another program called the *the Java Virtual Machine (JVM)*.

You execute the bytecode by an invocation like

```
java AddArray
```

A JVM executes the bytecode either in the interpreted mode, or by first converting the bytecode into a machine dependent executable by a second round of compilation known as *just in-time compilation (JIT)*.

JIT is the default mode. (It results in a 10-fold increase in execution speed over the interpreted mode.)

Before you can compile your Java program, you may have to tell the system how to go about locating your source file and the class file.

The default is your current directory.

But if your program uses code in a different directory, you have to tell both `javac` and `java` how to locate those classes.

The preferred to this is to use the `-classpath` option for both `javac` and `java`

```
javac -classpath .:directory_1:directory_2  sourceCode.java
```

```
java  -classpath .:directory_1:directory_2  className
```

On a Windows machine:

```
javac -classpath .;directory_1;directory_2 source.java
```

```
java -classpath .;directory_1;directory_2 className
```

If the classpath becomes too long, you can create a shell file on Unix/Linux machines and a bat file on Windows machines.

On Unix/Linux machines and on older PC platforms, you can also set the CLASSPATH environment variable.

If you are using either csh or tsch shells, you can include in your .cshrc file:

```
setenv CLASSPATH dir_1:dir_2:....
```

If you are using either sh or ksh, you can achieve the same effect by including the following strings in your *.profile* file:

```
CLASSPATH=dir_1:dir_2:....  
export CLASSPATH
```

Using the `-classpath` option overrides the environment variable.

Using either overrides the default, which is the current directory.

```

/* TermIO.c */

#include <stdio.h>

main()
{
    int i;
    int sum = 0;
    char ch;

    printf("Enter a sequence of integers:  ");

    while ( scanf( "%d", &i ) == 1 ) {
        sum += i;
        while ( ( ch = getchar() ) == ' ' )
            ;
        if ( ch == '\n' ) break;
        ungetc( ch, stdin );
    }
    printf( "The sum of the integers is: %d\n", sum );
}

```

```

//TermIO.cc
#include <iostream>
using namespace std;

int main()
{
    int sum = 0;
    cout << "Enter a sequence of integers: ";
    int i;
    while ( cin >> i ) {                                // (A)
        sum += i;
        while ( cin.peek() == ' ' ) cin.get();          // (B)
        if ( cin.peek() == '\n' ) break;                // (C)
    }
    cout << "Sum of the numbers is: " << sum << endl;
}

```

```

//TermIO.java

import java.io.*;

class TermIO {

    static boolean newline;                                // (A)

    public static void main( String[] args ) {
        while ( newline == false ) {
            String str = readString();                    // (B)
            if ( str != null ) {
                int data = Integer.parseInt( str );      // (C)
                System.out.println( "Integer entered: " + data );
            }
        }
    }

    static String readString() {                          // (D)
        String word = "";
        try {
            int ch;
            while ( ( ch = System.in.read() ) == ' ' )  // (E)
                ;
            if ( ch == '\n' ) {                          // (F)
                newline = true;                          // (G)
                return null;                             // (H)
            }
            word += (char) ch;                            // (I)
            while ( ( ch = System.in.read() ) != ' '    // (J)
                && ch != '\n' )
                word += (char) ch;                      // (K)
            if ( ch == '\n' ) newline = true;          // (L)
        } catch( IOException e ) {}
        return word;                                    // (M)
    }
}

```

}