Preliminaries

You must do this prior to creating and compiling any C++ programs.

Before compiling any programs, you will need some C++ files in your account. These are files that allow you to use the boolean type `bool` and the string type `apstring`. As you will discover in the lab, the ANSI C++ standard calls for the type `bool` and the type `string` but currently no compilers are fully compliant with the standard. We define the bool type and a string type called `apstring` in some special files. This allows you to write programs in the correct C++ style (except you use `apstring` instead of `string`.)

Log on to a Unix system and give the following commands in a terminal window

```bash
cd
mkdir c++
cd c++
```

The last command copies all files that end in either C or h to the local directory.

We will work in the c++ subdirectory of your home directory when we do Tasks 1, 2, and 3.

Background

Computer languages are designed to make it easy to write programs in a format that is formal yet English-like. There are literally hundreds of languages in everyday use. A few well-known languages in order of age are:

- FORTRAN (mid-1950’s) - mainly scientific use.
- COBOL (1958) - mainly business use.
- LISP (1958) - mainly use in artificial intelligence research.
- ALGOL 60 (1960) - first general purpose language.
- BASIC (1965) - designed to teach beginners how to program.
- Pascal (1970) - mainly used for teaching using structured programming techniques (based on ALGOL.)
- C (1971) - used in most operating systems programming. Allows access to the computer hardware while being very efficient as a general purpose programming tool. Standardized as ANSI C in the early 1990’s.
- Ada (1980) - developed by the Department of Defense as the single language which would be used by all their contractors (based on ALGOL and Pascal with many more features.)
- C++ (1984-1995) - evolved from C by adding classes. Designed for object-oriented pro-
Java (1995) - a pure object-oriented language developed by Sun Microsystems. The original uses were for programming applications that would be delivered via the World-Wide Web.

Many languages have developed that are variants of those above. For example, Visual Basic is a language derived from BASIC with graphics support added. It is useful for quick prototyping of user interfaces. The final interface is then developed in C, C++, or Java.

The types of problems being solved by the programs often determine whether one language is better suited for the program than another language. Both Java and C++ are relatively new languages that are suitable for solving a wide variety of problems. They both are object-oriented languages. This means that they fully support classes and objects.

If classes are removed, both languages have syntax like the C language. C was developed at AT&T Bell Labs (now Lucent Technologies) in the early 1970’s and became the most common programming language for operating system software. In addition, C was used for many application programs like word processors and spread sheets.

In the mid-1980’s object-oriented programming starting growing in importance as a way to speed software development. At Bell Labs a computer scientist, Bjarne Stroustrup, developed extensions to C that allowed the definition and use of classes. By 1990 this had evolved to the C++ language. The language became ‘standardized’ in 1995 as ANSI C++, but many older compilers do not yet implement the full standard. The fully compliant compiler on our Unix systems is called g++ (the Gnu C++ compiler). The version of the Sun C++ compiler (CC) that is currently installed on our systems is not fully compliant with the standard (for example, it doesn’t support the bool and string types).

In May, 1994 Sun Microsystems unveiled Java, a language it had been developing for several years for programming small appliances. Java has suddenly emerged as a possible replacement for C++ in some applications. Java is an interpreted language, which means that Java programs are compiled into a form that is independent of the type of computer on which the program will run. If a computer has a Java interpreter, it can run the program. Browsers like Netscape have a Java interpreter built in. This means that a compiled Java program can be downloaded over the Internet and run on the local computer using Netscape’s Java interpreter.

The developers of C++ and Java took advantage of much research on programming language design and both languages are quite powerful general purpose tools. C++ is somewhat more complex because it had to allow many of the features of C. These features are often difficult for beginners to understand and master. Java was a brand new language (although clearly influenced by C++) and hence was able to avoid some of the problem areas of C++.

**Java and C++: The Basics**
**Primitive Data Types**

The numeric types are the same in both languages, except Java specifies the range of values for a numeric type on every computer while the range may vary in C++. The character type char in Java is saved as a unicode value that satisfies an international standard and supports non-English alphabets. In C++ the char type is an 8-bits value. For ASCII characters (including characters formed on typical PC keyboards in the USA) the two languages agree.

### Table 1: Basic Data Types

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>int</td>
</tr>
<tr>
<td>float</td>
<td>float</td>
<td>float</td>
</tr>
<tr>
<td>double</td>
<td>double</td>
<td>double</td>
</tr>
<tr>
<td>char</td>
<td>char</td>
<td>char</td>
</tr>
<tr>
<td>boolean</td>
<td>bool</td>
<td>boolean</td>
</tr>
</tbody>
</table>

String Data

The type of a string is simple in Java. 'String' is the name of a provided Java class representing a block of char items.

**In C++ there are two types of strings.**

For historical reasons, any array of characters containing the special character ' \0' (called the null character) is considered a string. We will call this a C-style string. It really is just a special array. A typical declaration and initialization is

```
char *name = "Bob";  // the null character is implicit in "Bob"
```

or

```
char name[20];  // must be big enough
strcpy(name,"Bob");
```

These C-style strings come with a set of library functions that allows one to work with them. The strcpy function is one of these library functions. The asterisk in the first declaration indicates that name is a “pointer” to a block of characters. Pointers are discussed in the ‘Advanced’ section of the laboratory.

Standard C++ has a real `string` type that is a class. A typical declaration and initialization might look like:

```
string name("Bob");
```

The equivalent Java code would be

```
String name = "Bob";
```

or

```
String name = new String("Bob");
```

The second Java version creates a new object that is a clone of the string literal "Bob" and makes name a reference to it. The first Java version simply makes name a reference to the string literal "Bob". The C++ code is equivalent to the second Java version.
Operations

In Java and C++ the basic numeric operations are the same (+, *, /, %). The comparison operations (==, <, >, <=, ... etc.) are the same. The boolean operations (&&, ||, !) are the same.

Note that the new ANSI C++ standard allows use of the keywords and, or, and not as replacements for &&, ||, and !. This is a recent addition to the standard, and most compilers will not yet recognize them. (This includes g++.)

Statements

In Java and C++ the following statements are exactly the same.

- assignment
- while loop
- for loop
- if...else
- break
- switch

Conditional Expressions

In Java the conditions in loops and if statements must be of type boolean. In C++ they can be of any numeric or bool type. If a numeric expression appears where one expects a true or false value, C++ assumes that a zero value is false and any non-zero value is true!

Global Functions and Variables

Java forces all functions and variables to be within classes. C++ allows functions and variables to exist outside of any class. These are called global functions and variables and can be used by any other functions or classes. When Java wants to make a function available to other classes without creating an instance of the class containing the function it simply declares it "public static". Below are examples showing the use of main(). Notice that in the Java program the function main must be inside a class, although it is classified as "public static." We have underlined the parts of the Java code that would be stripped away to create a C++ program.

In Java main returns nothing (note the "void"), but in C++ it must return an integer. The value 0 signals successful completion of the program to the operating system.

In Java main must list its single parameter args which is an array of strings corresponding to any command line arguments to the program. In C++ this is optional unless the program will use the command line arguments. The ’Advanced’ section shows an example of how this is done in C++.

Arrays

Arrays in C++ and Java are similar, but C++ has no data value "length" associated with an array. The user must know explicitly what is the size of the array. The code below shows the declaration of an array.

---

<table>
<thead>
<tr>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class SmallestLegal</td>
<td>int main()</td>
</tr>
<tr>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>public static void main(String[] args)</td>
<td>return 0;</td>
</tr>
<tr>
<td>;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

---

Table 2: Minimal Program
of 200 integers and then the filling of each cell with the value 1.

Table 3: Array Use

<table>
<thead>
<tr>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int[] nums = new int[200];</code> // int nums[] = new int[200]; also works</td>
<td><code>int nums[200];</code></td>
</tr>
<tr>
<td>for (int i=0; i &lt;= nums.length-1; i++) {</td>
<td>for (int i=0; i &lt;= 199; i++) {</td>
</tr>
<tr>
<td>nums[i] = 1;</td>
<td>nums[i] = 1;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

Notice that C++ uses a very short form to declare the array of size 200. "nums.length" is not defined in C++.

Console Output

Both Java and C++ use the model of a "stream" of input from the keyboard and of output to the screen at the console. However, the format of the expressions is quite different.

Table 4: Console Output to Screen

<table>
<thead>
<tr>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>System.out.println(&quot;x is &quot; + x);</code></td>
<td><code>#include &lt;iostream.h&gt; // at beginning</code></td>
</tr>
<tr>
<td><code>System.out.print(&quot;Enter a value: &quot;);</code></td>
<td><code>// of program</code></td>
</tr>
<tr>
<td></td>
<td><code>cout &lt;&lt; &quot;x is &quot; &lt;&lt; x &lt;&lt; endl;</code></td>
</tr>
<tr>
<td></td>
<td><code>cout &lt;&lt; &quot;Enter a value: &quot;;</code></td>
</tr>
</tbody>
</table>

The name `cout` signals console output and the `<<` operation signals “insertion” of what follows into the stream (and hence onto the screen.) The special identifier `endl` stands for the insertion of a carriage return.

Console Input

In Java input from the keyboard is not easy. Most introductory courses use several special classes provided by the textbook author to hide the trickiness. (You may have used the ccj or BreezyGUI packages for this purpose, depending on when you took CS150.) In C++ reading values from the keyboard is very easy. We give some examples below. The `cin` combines with the extraction operation `>>` to take values from the keyboard. It automatically examines where the value is to be stored to determine its type.

Table 5: Console Input from Keyboard

<table>
<thead>
<tr>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>import ccj.*; // access special</code></td>
<td><code>#include &lt;iostream.h&gt;</code></td>
</tr>
<tr>
<td><code>.....</code></td>
<td><code>.....</code></td>
</tr>
<tr>
<td><code>int x;</code></td>
<td><code>int x;</code></td>
</tr>
<tr>
<td><code>String name;</code></td>
<td><code>char *name = new char [75];</code></td>
</tr>
<tr>
<td><code>.....</code></td>
<td><code>.....</code></td>
</tr>
<tr>
<td><code>x = Console.in.readInt();</code></td>
<td><code>cin &gt;&gt; x;</code></td>
</tr>
<tr>
<td><code>.....</code></td>
<td><code>.....</code></td>
</tr>
<tr>
<td><code>name = Console.in.readWord();</code></td>
<td><code>cin &gt;&gt; name;</code></td>
</tr>
</tbody>
</table>

We have used a C-style string in our example. Such a string is an array of characters with the null character (‘\0’) terminating it. We must specify the size of the array. The 75 characters specified must include space for the null character.
The compile commands and the run commands both differ in the two languages. In the examples below we assume that the entire program is in a single file. In Java all program files must end in `.java` while in C++ there are several conventions: `.cpp`, `.cc`, `.C`. The first is common on PC C++ compilers such as Borland/Inprise C++ and Microsoft Visual C++. The other two are common on Unix computers.

Let’s examine the C++ commands. The compile command is `g++`. It takes the option `-o First` which specifies the name of the output file, i.e., the compiled file. (The first two letters are a minus oh not a minus zero!) We want our executable to be called `First`.

To run the program, one simply types the name of the executable file, i.e., `First`. Because Java is an interpreted language, it was necessary to run the Java interpreter with the command `java` followed by the name of the compiled class, `First`. C++ produces executable code for the current computer. The compiled program will run on all computers of the type on which is was compiled, but it will not run on other types of computers.

### Side-By-Side Example

The two programs on page 8 are Java and C++ versions of a simple leap year program. Read them before proceeding.

#### Prototypes

There is one new concept in the example. At the beginning of main in the C++ program a “prototype” of the function `isLeapYear` appears. A prototype has the format

```plaintext
  type-of-return-value   name ( types of parameters );
```

You may optionally give the names of the parameters, but only the types really matter. The names will be ignored.

In Java prototypes are not required for technical reasons having to do with it being an interpreted language, but in C++ it is necessary that either the definition of a function precede any use of the function, or that a prototype declaration for the function precede any use of the function. If we omit the prototype from this program, the compiler generates the following error message:

```
LeapYear.C:14: warning: implicit declaration of function `int isLeapYear(...)'
```

### Task 1

Have you performed the Preliminaries on page 1? If not do them now.

You will need to be in the correct subdirectory. In a terminal window type

```plaintext
pwd
```
You should see

/home/MCS-Stu2/netid/c++

where netid is replaced by your login. You may see MCS-Stu rather than MCS-Stu2. The key is the end of the command. If you don’t see this, then you are in the wrong directory and must enter the following commands:

```bash
cd
cd c++
```

Now you are ready to work!

**Read the programs carefully.** Then using xemacs create the C++ program LeapYear.C, compile it, and run it. Recall that the commands would be

```bash
xemacs LeapYear.C &
```
to create it,

```bash
CC -o LeapYear LeapYear.C
```
to compile it, and

```bash
LeapYear
```
to run it.

**Task 2**
Write a C++ program Reverse.C which will read in 10 integers from the keyboard and then display them one per line in the reverse order in which they were entered. This requires an array.

**Task 3**
Write a C++ program SortArray.C which will read in 10 integers from the keyboard into an array, then sort the array using the selection sort. Finally, print out the array, one value per line.
Advanced C++ Topics

There are a number of C++ topics that are somewhat special and handled differently from Java. We list them below for reference.

Reference Parameters

In Java all argument values are passed by value. This means that a copy of the value is used in the function or method. For primitive data types like int, double, char, this means that attempting to change the original values while within the function will fail. Consider the following Java program and guess what the output will be.

```java
public class BadSwap {
    public static void main(String[] args) {
    // Your code here
    }
}
```

Table 7: A Leap Year Example

<table>
<thead>
<tr>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>// LeapYear.java</td>
<td>// LeapYear.C</td>
</tr>
<tr>
<td>// Your name(s) here</td>
<td>// Your name(s) here</td>
</tr>
<tr>
<td>// Requests a year from the user and</td>
<td>// same comments as in Java program</td>
</tr>
<tr>
<td>// indicates whether or not it is a</td>
<td></td>
</tr>
<tr>
<td>// leap year. No error checking is</td>
<td></td>
</tr>
<tr>
<td>// done.</td>
<td></td>
</tr>
<tr>
<td>import ccj.*;</td>
<td></td>
</tr>
<tr>
<td>public class LeapYear</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>public static void main(String[] args) {</td>
<td></td>
</tr>
<tr>
<td>System.out.println(&quot;Enter a year&quot;);</td>
<td></td>
</tr>
<tr>
<td>int y = Console.in.readInt();</td>
<td></td>
</tr>
<tr>
<td>if (isLeapYear( y )) {</td>
<td></td>
</tr>
<tr>
<td>System.out.println(y + &quot; is a leap year&quot;);</td>
<td></td>
</tr>
<tr>
<td>} else {</td>
<td></td>
</tr>
<tr>
<td>System.out.println(y + &quot; is not a leap year&quot;);</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>// isLeapYear - returns true or false depending on parameter year</td>
<td></td>
</tr>
<tr>
<td>public static boolean isLeapYear(int year) {</td>
<td></td>
</tr>
<tr>
<td>if (year%400 == 0) return true;</td>
<td></td>
</tr>
<tr>
<td>else if (year%100 == 0) return false;</td>
<td></td>
</tr>
<tr>
<td>else if (year%4 == 0) return true;</td>
<td></td>
</tr>
<tr>
<td>else return false;</td>
<td></td>
</tr>
<tr>
<td>} // end of class LeapYear</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>// LeapYear.C</td>
<td></td>
</tr>
<tr>
<td>// Your name(s) here</td>
<td></td>
</tr>
<tr>
<td>// same comments as in Java program</td>
<td></td>
</tr>
<tr>
<td>#include &lt;iostream.h&gt;</td>
<td></td>
</tr>
<tr>
<td>int main() {</td>
<td></td>
</tr>
<tr>
<td>bool isLeapYear(int ); // prototype</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; &quot;Enter a year&quot; &lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>int y;</td>
<td></td>
</tr>
<tr>
<td>cin &gt;&gt; y;</td>
<td></td>
</tr>
<tr>
<td>if (isLeapYear( y )) {</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; y &lt;&lt; &quot; is a leap year&quot;</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>} else {</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; y &lt;&lt; &quot; is not a leap year&quot; &lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>} return 0;</td>
<td></td>
</tr>
<tr>
<td>// isLeapYear - ....</td>
<td></td>
</tr>
<tr>
<td>// bool isLeapYear(int year) {</td>
<td></td>
</tr>
<tr>
<td>if (year%400 == 0) return true;</td>
<td></td>
</tr>
<tr>
<td>else if (year%100 == 0) return false;</td>
<td></td>
</tr>
<tr>
<td>else if (year%4 == 0) return true;</td>
<td></td>
</tr>
<tr>
<td>else return false;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

```
{  
    int x = 5;
    int y = 8;
    swap(x,y);  // attempt to exchange their values
    System.out.println("x= " + x + " and y= " + y);
}

public static void swap(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}

The output will be
   
x= 5 and y= 8
showing that the swap routine didn’t change the values of x and y. What happened was that the parameter a was given a copy of the value in x, i.e. 5, and the parameter b was given a copy of the value in y, i.e. 8. Then a and b were exchanged, but it had no effect on x and y. Simply put, the method swap did not have access to the values that corresponded to a and b. It couldn’t get to x and y!

This is only a problem for primitive types in Java. Array names are references to the original block of cells. Any variables that refer to object values from a class are true references and the parameter is a reference to the original object.

How does C++ fix the swap problem? C++ allows reference parameters. In C++ a properly working swap method for integers is shown below

```cpp
void swap( int& a, int& b)  
{  
    int temp = a;
    a = b;
    b = temp;
}
```

Notice that the types of the parameters are int& which means that instead of getting copies of the variables to be exchanged, a and b will actually refer to the originals!

Note that int& a can be written int &a.

**Pointers**

A pointer is a reference variable. This means that its value is a reference to data somewhere in memory. **Rather than holding the data itself, it holds the address of where the data can be found.** We illustrate the notation and simple use with a partial example.

```cpp
int *p;  // p will hold an address of an int value
```
int x = 5;
p = &x;  // &x means address of x
cout << "x= " << *p << endl; // in this context, *p means
    // go to the address that p holds
    // and get the value there (5)
    
cout << "x= " << x << endl;

The previous code will produce

x= 5
x= 5

All array names are really pointers to the first cell in the block of cells that is the array.

The string Type

The use of the string type is illustrated by a program which will read in 3 ‘words’ and produce the concatenated value.

    // StringExample.C
    // Reads in 3 words and produces the concatenated result.
    #include <iostream.h>
    #include <string>

    int main()
    {
        cout << "Enter 3 words please" << endl;
        string result("" ); // creates an empty string
        string word;       // says word will hold a string
        for (int i=1; i<=3;i++)
        {
            cin >> word;
            result = result + word; // concatenation operation is +
        }
        cout << "Concatenation is "" " " result << "\"" << endl;
        return 0;
    }

    Here is the compile command for this program.

    CC -o APStringExample  APStringExample.C  apstring.C

    The program is run by typing StringExample.

Operator Overloading

Java normally does not allow overloading of operators except in the natural way for primitive types. What does overloading mean? It simply means that the operator can be used with many types. For example the addition operator + can be used to add int, double, float types. It also is overloaded to mean concatenation when used with String types. The user cannot define + on classes. This is what is meant by not allowing overloading. For example, if name1 and name2 are String variables in Java, name1 == name2 is not a comparison of the string values because == is not overloaded. We had to use name1.equals(name2) to compare for equality.

C++ allows the user to define all typical operators on classes where it might make sense. In the string class, == is defined to compare two strings. Many other operators like < and > and <= are also defined.
C-Style Strings and the Library

Recall that C-style strings are really just arrays of char containing the null character. We repeat the previous example using this older string model. We make use of functions provided in the string library for working with these C-style strings.

```c
// CStyleExample.C
// Reads in 3 words and produces the concatenated result.
// Uses the C-style string convention where a string is an
// array of characters with the null character marking its end.
#include <iostream.h>
#include <string.h> // note: This accesses the C-style
      // string library operations
int main()
{
  cout << "Enter 3 words please" << endl;
  char result[301]; // we allocate space for 3 100 character words
  result[0] = '\0'; // makes result be the empty C-style string
  char word[101];
  for (int i=1; i<=3;i++)
  {
    cin >> word;    // reads a 'word' into word AND inserts the
      // null character
    strcat(result,word);  // does result= result+word
  }
  cout << "Concatenation is " << result << endl;
  return 0;
}
```

The compile command is simple as the C-style string library is part of the compiler and is automatically compiled.

```
CC -o CStyleExample CStyleExample.C
```

Command Line Arguments in C++

The following program will print out its command line arguments one per line. It is important to note that argv[0] is the command itself. This is different from Java. Furthermore, since C++ doesn’t have a length data value for arrays, the number of elements in the array argv must be provided. This is what argc is. Normally argc means "arg count" and argv means "arg values".

```c
// PrintArgs.C
// Displays each command line argument to the screen,
// one per line.

#include <iostream.h>

int main(int argc, char *argv[])
{
```
for (int i=0 ;i < argc ; i++ )
{
    cout << argv[i] << endl;
}

File I/O

The following C++ program will open the file Grades.dat which holds a list of floating point values and compute the average.

// AverageFile.C
// Opens the file Grades.dat and reads in floating point values until
// the end of file, then displays the average.
#include <fstream.h>    // note change
#include <stdlib.h>

int main()
{
    ifstream infile("Grades.dat");
    if (infile == NULL)
    {
        cerr << "Unable to open Grades.dat" << endl;
        exit(0);  // quits program immediately
    }
    double sum = 0.0;          // used for running total
    int n = 0;                 // used to count values
    double val;
    infile >> val;             // just like the use of cin
    while (!infile.fail())     // check for problems
    {
        sum = sum + val;
        n = n + 1;
        infile >> val;          // get next value
    }
    infile.close(); // good practice but not essential
    if (n == 0)
        cout << "No values found in file" << endl;
    else
        cout << "Average = " << sum/n << endl;
    return 0;
}

Summary

C++ and Java are quite similar. C++ has a few more specialized features. A good C++ text is all that is needed to master the language. The comments above are to help with the transition.