C and C++ Input/Output for the ACM International Computer Programming Contest

Full C/C++ input/output can be tricky. Fortunately, the ICPC programming problems all seem to use a fairly simple data file format philosophy:

- Everything seems to be freeform input, that is, there are no fixed columns where the data must reside.

- The slight exception to this is that there appears to be instances of a complete string on a line by itself. Sometimes, this string is allowed to have whitespace in it, which must be read as being part of the string.

- Output for the judging process looks like it needs to be printed freeform, typically with one data value per line

- A printed line ends with a newline, “\n”, not a “\r\n”

- All input comes in on standard input (cin, stdin).

- All output goes to standard output (cout, stdout).

- In the instructions for the problem, they always tell you what the maximum number of characters a string will contain. This means that you can statically dimension that character array, remembering to include at least one extra character for the ‘\0’.

So, here’s what you need to know

Lines to always include:

At the top of your C program, always include the lines:
```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
```

At the top of your C++ program, always include the lines:
```
#include <math>
#include <cctype>
#include <string>
using namespace std;
```

This will save you hassles later.

Assuming you have these data types . . .
```
int    i;
float  f;
double d;
char   c;
char   carray[256];     // a list of characters, terminated with a NULL, ‘\0’
```

To read scalar variables, just extract them from standard input:

In C:

```
      scanf( “%c”,  &c );
      scanf( “%d”,  &i );
      scanf( “%f”,  &f );
      scanf( “%lf”, &d );
```

In C++:

```
      cin >> c;
      cin >> i;
      cin >> f;
      cin >> d;
```

To read a character array up to the next whitespace (blank, tab, \n), do this:

In C:

```
      scanf( “%s”, carray );
```

In C++:

```
      cin >> carray;
```

These will automatically add the NULL character at the end of the string for you.

To print a scalar variable, do this:

In C:

```
      printf( “%c\n”,  c );
      printf( “%d\n”,  i );
      printf( “%f\n”,  f );
      printf( “%lf\n”, d );         // “lf” stands for “long float”
```

In C++:

```
      cout << c << endl;            // “endl” stands for “end line”, i.e. \n
      cout << i << endl;
      cout << f << endl;
      cout << d << endl;
```

To print an array of characters, do this:

In C:

```
      printf( “%s\n”, carray );
```

In C++:

```
      cout << carray << endl;
```

The C++ *string* Object

The C++ language has a special *string* object to hold an array of characters. The advantage of using it instead of an array of characters is that it also includes several built-in methods that do handy-dandy string manipulation for you (built-in functionality is always a good thing in the heat of a competition)

Assume we have variables:

```
#include <string>
string  str, str1, str2;
char    c;
string::size_type   pos;
```

To read a string that has no whitespace in it, do this:

```
cin >> str;
```

To read a string that has whitespace in it and is on a line by itself, do this:

```
getline( cin, str );
```

Here's a good web site on C++ file i/o methods:

http://www.cplusplus.com/doc/tutorial/files/

Print this out before the competition!

Always, Always, Always:

Echo the input right after you read it so that you can be sure you are reading it correctly! You'd be surprised how much time gets wasted debugging an algorithm when it was actually the input that was wrong.

Print these lines to standard error (cerr, stderr) so that they do not get into your output stream:

In C:

```
fprintf( stderr, "Echo: i = %d\n", i );
```

In C++:

```
cerr << "Echo: i = " << i << endl;
```

No, No, No: don't wait until you have a debugging issue. *Do this right from the start.*

Comment-out those print lines before submitting the code for judging.

Testing Your Program:

Type in the sample input file that was given in the problem. Call it something like `testinput.txt`

Once your program compiles, run it on the command line like this:

```
prog  <  testinput.txt  >  testoutput.txt
cat  testoutput.txt
```

Your standard error read-echoing messages will show up on the console while your program is running. You can then look at `testoutput.txt` to see if it matches what you think your algorithm should be producing.

An Example

For example, the first problem in 2014’s Division 2 gives a sample input file that looks like this:

```
2
8 12
4 6
```

And a sample output of:

```
6
4
```

To do this in C:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
int main( )
{
      int numTests, v, e, t, f;
      scanf( “%d”, &numTests );
      fprintf( stderr, “Echo: numTests = %d\n”, numTests );
      for( t = 0; t < numTests; t++ )
      {
            scanf( “%d %d”, &v, &e );
            fprintf( stderr, “Echo: v = %d, e = %d\n”, v, e );
            f = 2 - v + e;
            printf( “%d\n”, f );
      }
      return 0;
}
```

To do this in C++:

```
#include <math>
#include <cctype>
#include <string>
using namespace std;
int main( )
{
      int numTests;
      cin >> numTests;
      cerr << “Echo: numTests = ” << numTests << endl;
      for( int t = 0; t < numTests; t++ )
      {
            int v, e;
            cin >> v >> e;
            cerr << “Echo: v = “ << v << “, e = “ << e << endl;
            int f = 2 - v + e;
            cout << f << endl;
      }
      return 0;
}
```

Here are C++ *string* class methods you might find useful:

String assignment	`str = str1;`
String concatenation	`str = str1 + str2;`
String concatenation	`str += str1;`
Character concatenation	`str += c;`
Length	`int len = str.length( );`

Accessing a single character	`char c = str[10];`
	`str[10] = c;`

String equality	`if( str == str1 )  . . .`
Alphabetical order testing	`if( str > str1  &&  str < str2 ) . . .`
Finding a character, starting at the front	`string::size_type i = str.find(  ‘x’ );`
Finding a character, starting at the end	`string::size_type i = str.rfind( ‘x’ );`
Finding a character, starting at pos	`string::size_type i = str.find(  ‘x’, pos );`
Finding a character, starting at pos	`string::size_type i = str.rfind( ‘x’, pos );`
Finding one character of a string	`string::size_type i = str.find(  “abc” );`
Finding one character of a string	`string::size_type i = str.rfind( “abc” );`
Finding one character of a string, starting at pos	`string::size_type i = str.find( “abc”,  pos );`
Finding one character of a string, starting at pos	`string::size_type i = str.rfind( “abc”, pos );`
Value signifying nothing was found	`if( i == string::npos ) . . .`
Extracting a sub-string	`str2 = str.substr( pos, len );`
Replacing a sub-string	`str.replace( pos, len, “newstr” );`
Erasing a sub-string	`str.erase( pos, len );`

Here’s a good web site for more detail on C++ *string* methods:

http://www.mochima.com/tutorials/strings.html
Print this out before the competition!

Use doubles, not floats?

We have been told by other teams to always use doubles, not floats, so that you don’t run out precision for your answers. We don’t have any experience with this, but this is what we have heard.

Another Piece of Advice

Don’t forget the

```
    return 0;
```

at the end of your main program. Apparently the judging software often (always?) looks for that.

In Case You *Do* Need to Open a File in C:

Just in case they throw you a curve and ask you to read from a file, and you want to use C, do this:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>

int main( )
{
      char *filein  = “filetoread.txt”;
      char *fileout = “filetowrite.txt”;
      int i;

      FILE * reader;
      FILE * writer;

      reader = fopen( filein,  “r” );
      if( reader == NULL  )
      {
            fprintf( stderr, “Cannot open file %s\n”, filein );
            return 1;
      }

      writer = fopen( fileout, “w” );
      if( writer == NULL  )
      {
            fprintf( stderr, “Cannot create file %s\n”, fileout );
            fclose( reader );
            return 1;
      }

      fscanf( reader, “%d”, &i );
      i = i + 1;
      fprintf( writer, “%d\n”, i );

      fclose( reader );
      fclose( writer );

      return 0;
}
```

In Case You *Do* Need to Open a File in C++:

Just in case they throw you a curve and ask you to read from a file, and you want to use C++, do this:

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

int main( )
{
      string filein  = “filetoread.txt”;
      string fileout = “filetowrite.txt”;;
      int i;

      ifstream reader;
      ofstream writer;

      reader.open( filein,  ios::in );
      if(  ! reader.is_open( )  )
      {
            cerr << “Cannot open file “ << filein << endl;
            return 1;
      }

      writer.open( fileout, ios::out );
      if(  ! writer.is_open( )  )
      {
            cerr << “Cannot create file “ << fileout << endl;
            reader.close( );
            return 1;
      }

      reader >> i;
      i = i + 1;
      writer << i << endl;

      reader.close( );
      writer.close( );

      return 0;
}
```