Introduction to Programming Contests via C++

The goal of these course notes is to introduce students who have a light introduction to programming in any traditional language (Python, Java, etc.) to C++ syntax via programming competition problems using the platform Kattis. While general introductory programming courses focus on program design, user interfaces and many facets of programming, competitions tend to put sole focus on the problem solving aspect of programming by minimizing user interfaces and creating short, to the point exercises that truly focus on problem solving skills. This approach isn't for everyone, and if this exposure is one's only exposure to programming, then there are many skills necessary to learn for "real world" programming that aren't covered in these notes. Nor are these notes meant to be close to any sort of comprehensive guide on C++ syntax. Rather, only aspects of the syntax that tend to be necessary for programming competitions is included. This allows students to dig into the problem solving aspect of programming much faster, and is the primary aim of these notes. (Note: Other comparable notes might have a WHOLE chapter setting up the purpose of programming, without actually getting into any programming!)

Kattis Platform

Kattis is a website that runs many programming competitions and archives problems from those competitions for users to solve. The URL for the website is:

https://open.kattis.com/

When you get to the website, the first two tabs you see are "Contests" and "Problems." We will use both of these tabs during this course. The website allows you to click on a problem, read it, create a program to solve the problem and submit your solution. The website will give you back one of the following responses:

Accepted (yeah! your program correctly solved all of the judge's test cases!)
Wrong Answer (sorry, your program produced the wrong output on at least one test case.)
Time Limit Exceeded (sorry, on one of the judge's test cases, your program took too long.)
Run Time Error (sorry, on one of the judge's test cases, your program crashed.)
Compile Time Error (sorry, your submitted source code does not compile.)
Output Limit Exceeded (sorry, your program created too much output.)
Memory Limit Exceeded (sorry, your program used too much memory.)

A complete description of these responses is given here:

https://open.kattis.com/help/judgements

Standard Output

The most simple thing a computer program can do is print out a fixed message. Here is a simple C++ program that prints out the traditional phrase, "Hello World!":

```
// Arup Guha
// 12/20/2022
// Hello World! for SI@UCF Camp Notes

// Useful to reduce typing and allow us to do basic input/output.
using namespace std;
#include <iostream>

// Part of every C++ program.
int main() {

    // This is the way to output a message to the screen.
    // Each item is separated with <<
    cout << "Hello World!"  << endl;

    // Last line of every main function.
    return 0;
}
```

In fact, you should try it out. Type up this program and submit it here:

https://open.kattis.com/problems/hello

All the lines that start with // are comments. These are ignored by the compiler and just have information for other humans reading the program, in this case, new students learning C++.

In order to avoid typing "std::" many times, for competitive programming, it's advised to use the namespace std.

In order to use functions built into C++, you have to include various libraries. Eventually we'll use a trick to include many libraries at once, but for learning purposes, at first, as you learn new functions, the libraries from which they come will be delineated, one by one. To write output to the screen in C++ the library iostream must be used. It allows you to use the cout object which makes it easy to produce standard output.

Every C++ program must have a main function. The main function is the only one that automatically executes. For real world programming, a program may have thousands of functions, but for competitive programming, most programs will just have a few functions, at most. For simple beginning programs, we'll only use a main function. The first and last lines of main will be the same every time for our purposes. An open brace ('{') denotes the beginning of a function or block of code and a close brace ('}') denotes the end of a function or block of code.

To use the cout function to output text, put cout, followed by "<<", then the item you want to print. You can have as many "<<" item pairs as you want. A string literal is denoted in C++ with a pair of double quotes. Nearly everything between a pair of double quotes will print exactly. For competitive programming, the output formats tend to be quite simplified, so these notes won't go

into nearly as much detail as regular notes do about output. Soon, we'll learn how to output the value of a variable, and this will be the primary form of output in most competitive programming problems. Finally "endl" is short for an endline character. This means that the if another cout would occur, it would start on the following line.

Finally, you'll notice that all statements in C++ end in a semicolon. It'll see like there are exceptions to this rule, but in reality, there are not.

Variables and Types

In all programming languages, data is stored in variables. Every language has some types (of variables) that are native (part of the language automatically. For C++ the types we'll use frequently are as follows:

bool – variables must have the value true or false.

char – the character type, a single character can be stored in a char variable.

int – integers in between about negative and positive 2 billion can be stored in variables of this type. Many beginning programs with integer numerical data should use this type, but for some problems, this type doesn't suffice because the problem requires storage of integers greater than the range of int.

long long – integers in between about -8×10^{18} and $+8 \times 10^{18}$, rarely do competitive programming problems require storage of integer information outside of these bounds.

double – allows the storage of real numbers (with decimals) with a relative precision of about 12 digits.

In order to use a variable in C++, it must first be declared with its type. Here is an example:

```
int sum = 0;
```

While it's not required to set a variable to a value when you declare it, it's a good practice because if you don't, any random value can be assigned to the variable initially.

Standard Input

In a regular C++ program, usually, the program prints out a message to the user, asking the user to enter information. In competitive programming, this step is skipped. Your program is expected to read in information based on a specification without being prompted to do so.

In order to read in information into a variable, do the following:

```
cin >> var;
```

where var is the name of the variable. After this statement executes, whatever information was entered from standard input gets stored in the variable named var. We'll see an example shortly.

Furthermore, if you have multiple pieces of information to read in, you can chain them together in a single cin as follows:

```
cin >> var1 >> var2 >> var3;
```

In competitive programming, you don't have to worry about this too much, but each piece of information must be separated by white space. (That's how the computer knows what to store in var1, var2 and var3, respectively.) If the information entered doesn't match types of the variable declarations, then this can crash your program. So, it's important to make sure you read in information into variables of the appropriate type.

Assignment Statement

All programming languages have an assignment statement, so we won't go into too much detail here, but the general form is as follows:

```
var = expression;
```

The statement first evaluates the value of the expression on the right hand side, then it sets the value of the variable on the left hand side equal to this value.

For example, the output of the following set of statements:

```
int a = 1, b = 2, c;

c = a + b;
cout << a << " " << b << " " << c << endl;
a = b;
cout << a << " " << b << " " << c << endl;
b = c;
cout << a << " " << b << " " << c << endl;
```

is

```
1 2 3
2 2 3
2 3 3
```

Our Second Kattis Program: R2

Here is a link to the problem description:

https://open.kattis.com/problems/r2

Thus, we must design a program that reads in two variables, a number followed by the average of that first number and a second unknown number, and then we must figure out that unknown number and output it.

It should be fairly easy to see that whatever the difference is between the first two input terms (second term minus first term), we take that difference and add it to the second term to obtain our answer.

So, for their first sample, we have 15 – 11 = 4, thus the third term is 15 + 4 = 19. This is consistent with the given problem since the average of 11 and 19 is indeed 15.

For their second sample, we have 3 – 4 = -1, thus the third term is 3 + (-1) = 2.

Thus, for our solution, we'll do the following:

1) Read in the given information into variables a and b.

2) Calculate the difference b – a and store this in a variable, diff.

3) Output b + diff.

Though this program is fairly easy, it's still good to outline our steps on paper first before coding. Here's the program. Internal comments have been removed for brevity. The posted file has those comments.

```
// Arup Guha
// 12/20/2022
// R2 for SI@UCF Camp Notes
// https://open.kattis.com/problems/r2

using namespace std;
#include <iostream>

int main() {

    int a, b;
    cin >> a >> b;

    int diff = b - a;
    cout << b + diff  << endl;

    return 0;
}
```

One More Example with Basic Arithmetic: Jack-O'-Lantern Juxtaposition

Here's a link to the problem:

https://open.kattis.com/problems/jackolanternjuxtaposition

Once you read through the whole story, you realize that all you have to do is read in three integers and output their product. Here's the program!

```
// Arup Guha
// 12/20/2022
// Jack-O'-Lantern Juxtaposition for SI@UCF Camp Notes
// https://open.kattis.com/problems/jackolanternjuxtaposition

using namespace std;
#include <iostream>

int main() {

    // Short and sweet!
    int a, b, c;
    cin >> a >> b >> c;
    cout << a*b*c << endl;

    return 0;
}
```

Arithmetic Expressions: Integer Division and Mod

All students learn PEMDAS in mathematics class and learn the proper order of operations between multiplication, division, addition and subtraction. These rules also work for C++ expressions. Two C++ arithmetic operations that students might not familiar with are:

(1) integer division
(2) modulus

The first is very tricky because it uses the same symbol as regular division: /.

The way the computer determines whether to do a regular division or a integer division is by looking at the types of both operands. If both are integer types (int or long), then an integer division is performed. If either is a double, then a real number division is performed.

Here is the definition of integer division in C++.

```
a/b
```

If the real division result is positive, the integer division is the greatest integer less than or equal to the real division result.

If the real division result is negative, the integer division is the least integer greater than or equal to the real division result.

Otherwise, if a = 0 (we are already assuming b ≠ 0), the result is 0.

In practice, we usually deal with positive numbers, so the integer division for most problems represents the quotient of the division as taught in grade school, the number of full times one number divides into another. (So, if I have 43 piece of pizza to give to 6 people, I can guarantee that each person receives at least 7 slices.)

Also, on occasion, you'll want to do a real number division between two integer variables, a and b. The easiest way to accomplish this is `1.0*a/b`. The multiplication comes first, and C++ changes the working expression to a double since 1.0 is a double. Then the division becomes a double divided by an integer, for which a real number division is performed.

The operation that goes along with integer division is the modulus operation, which is denoted with the % operator: `a%b`

If both a and b are positive, this evaluates to the remainder when a is divided by b.

If a is negative and b is positive, this evaluates to `-((-a)%b)`, so it's the same as "pulling out the negative sign", evaluating the modulus with positive operators and then negating the answer.

If b is negative, this evaluates to `a%(-b)`, thus effectively, the sign of b doesn't matter and for the purposes of the operation, its absolute value is used.

Integer division and modulus are extremely useful for competitive programming. Note the following for an integer variable x storing a non-negative value:

`x%10` evaluates to the units digit of x.
`x/10` evaluates to the number you get when you chop off the last digit of x.
`x%2` is 0 when x is even, and 1 when x is odd.

Let's look at problem that utilizes integer division and mod.

Integer Division and Mod Example: Digit Swap
Here's a link to the problem:

https://open.kattis.com/problems/digitswap

Let us come up with an algorithm to solve the problem:

1) Read the input into an integer variable, num.
2) Extract the units digit of num via %.
3) Extract the tens digit of num via /.
4) Output the result in one of two ways:
    a) Output the two digits in reverse order, OR
    b) Calculate a single integer via the formula 10*(new tens digit) + (new units digit).

Here is the main function for both solutions:

```
int main() {

    // Get number and separate out two digits.
    int num;
    cin >> num;
    int unit = num%10;
    int ten = num/10;

    // Here we output each digit, but in reverse order.
    cout << unit << ten << endl;

    return 0;
}
```

```
int main() {

    // Get number and separate out two digits.
    int num;
    cin >> num;

    // Use the expression for the units digit and tens
    // digit to form the new number mathematically.
    cout << 10*(num%10) + num/10 << endl;

    return 0;
}
```

Real Number Division Example: Triangle Area

Here's a link to the problem:

https://open.kattis.com/problems/triarea

Here we can read in our height and base as doubles, or when we divide by 2 in our formula, divide by 2.0 forcing a real number division. Both solutions are included:

```
int main() {
    int height, base;
    cin >> height >> base;
    cout << height*base/2.0 << endl;
    return 0;
}
```

```
int main() {
    double height, base;
    cin >> height >> base;
    cout << height*base/2 << endl;
    return 0;
}
```