REVIEW OF C

Basic Data Types

- There are four basic data types in C: int, float, double, and char.
  - The int data type is intended to store whole numbers.
  - The float data type is intended to store real numbers.
  - The double data type is also intended to store real numbers but has twice the precision so that it can store a larger range of numbers.
  - The char data type is intended to store character symbols and controls used to display text.

- The following code demonstrates the differences between the types:

```
#include <stdio.h>
int main()
{
int x,y;
char a;
float f,e;
double d;
x=4;
y=7;
a='H';
f=-3.4;
d=54.123456789;
e=54.123456789;
printf("%d %c %f %lf\n",x,a,e,d);
printf("%d %c %.9f %.9lf\n",x,a,e,d);
}
```

- Executing this code produces the following result:

  4 H 54.123455 54.123457

  4 H 54.123455048 54.123456789
- In the first line of output, the float variable has seemingly been rounded downward in the last displayed digit, while the double variable has correctly been rounded upward in the last displayed digit. In fact, the float has simply run out of precision. This can be seen in the second line of output, where both variables are forced to print to nine decimal places.
- The double variable has the correct value, but the float has erroneous values in the latter digits.
- The printf() and scanf() functions are the primary output and input functions in C. They are included in the C standard library, which is usually linked to an executable by default.

Basic Arithmetic

- The basic arithmetic operations supported in C include addition, subtraction, multiplication, division, and modulus (remainder).
- The operators ++ and -- are provided for use within loops to increment (add 1 to) or decrement (subtract 1 from) a variable. this reason.

Consider the following code :

```
#include <stdio.h>
int main()
{
int x,y;
int r1,r2,r3,r4,r5;
x=4;
y=7;
r1=x+y;
r2=x-y;
r3=x/y;
r4=x*y;
printf("%d %d %d %d\n",r1,r2,r3,r4);
r3++;
r4--;
r5=r4%r1;
printf("%d %d %d\n",r3,r4,r5);
}
```

- The output of executing this code is as follows:

```
11 -3 0 28
1 27 5
```

- The modulus operator can be used only on integer variables. All the other arithmetic operators can be applied to all variables.

Loops

There are three basic types of loops in C: for, while, and do-while.

- The for loop is intended to be executed a fixed number of iterations, known before the loop is entered.Hence, it is given both a starting condition and an ending condition.
- The while loop is intended to be executed an unknown number of iterations. Hence, it is only given an ending condition.
- The do-while loop is also intended to be executed an unknown number of iterations but will be executed at least once.

NOTE: The while loop may be executed zero times if it fails the condition on the first attempt. The do-while loop does not test the condition until it has finished the loop, so it will execute the loop at least once.

The following code demonstrates all three types of loops:

```
#include <stdio.h>
int main()
{
int i,x;
x=0;
for (i=0; i<4; i++)
{
x=x+i;
printf("%d\n",x);
}
while (i<7)
{
x=x+i;
i++;
printf("%d\n",x);
}
do
{
x=x+i;
i++;
printf("%d\n",x);
}
while (i<9);
}
```

The following is the output of executing this code:

Conditionals and Blocks

- The basic conditional in C is the if-else statement. It supports tests for equality (==), inequality (!=), and relative size (>, <, >=, and <=).
- Multiple conditions can be tested within a single statement using the logical AND (&&) and logical OR (||) operators to group the individual conditions.
- Statements (individual lines of code) are grouped using brackets ({}). In the absence of brackets, a conditional or loop statement applies only to the single following statement.

The following code demonstrates conditionals and blocks:

```
#include <stdio.h>
int main()
{
int i,x;
x=0;
for (i=0; i<5; i++)
{
if (i%2 == 0 || i == 1)
x=x+i;
else
x=x-i;
printf("%d\n",x);
}
}
```

- The following is the output of executing this code:

```
0
1
3
0
4
```

Flow Control

- There are two flow control statements that change the way an interation through a loop is executed: continue and break.
- The continue statement returns control to the beginning of the loop, testing the loop conditional to start the next iteration. In effect, it skips the rest of the current iteration and starts the next one.
- The break statement terminates the loop and immediately proceeds to the next line of code following the loop.
- The following code demonstrates flow control:

```
#include <stdio.h>
int main()
{
int i,x;
x=0;
for (i=0; i<5; i++)
{
if (i%2 == 0)
continue;
x=x-i;
if (i%4 == 0)
break;
printf("%d\n",x);
}
}
```