while and for difference

while is used when you don't know how many time you have to repeat, so repeat WHILE condition is true.

for is used when you know FOR how many time you have to repeat.

When to use which loop ?

1. While and doWhile are similar, they are used when we don’t know number of times of repetitions

2. For is used when we know the number of iterations.

3. For each is used with array or STL.

What should be used in for loop i++ or ++i ?

Anyone can be used.

++i; is known to be faster than i++;.

Reason:

i++; is same as i=i+1; here i+1 is evaluated and the result is stored in a temporary variable and then assigned to i.

++i; here i+1 is not stored in a temporary variable, directly i is increased by 1

Factorial for large numbers

if the data type is taken long long then factorial will work upto 20!

if you want beyond 20! then take a string and write own procedure to calculate factorial.

Armstrong Number, why number is stored in m ?

actual number is taken in n.

While loop will stop when n=0

We have to compare sum with the number, so it is already stored in m.

Reverse a number ending with zero.

if a number is ending with 0 and you want 0 also then,

Example : n=2500

You have to store the reverse number in a string

```
char reverse[10];
int r,i=0;
while(n>0)
{
        r=n%10;
        reverse[i]=r+'0';  // '0' is added to make a digit as character.
        n=n/10;
}
reverse[i]='\0';
cout<<reverse;
```

Display a number in words.

if a number is ending with 0 and you want to display in words then,

Example : n=2500

You have to store the reverse number in a string

```
char reverse[10];
int r,i=0;
while(n>0)
{
        r=n%10;
        reverse[i]=r+'0';  // '0' is added to make a digit as character.
        n=n/10;
}
reverse[i]='\0';

for(int i=0;reverse[i]!='\0';i++)
{
        switch(reverse[i]-'0')
        {
                case 0: cout<<"Zero ";
                        break;
                case 1: cout<<"One ";
                        break;
                case 2: cout<<"Two ";
                        break;
                .
                .
                .
        }
}
```