The for Loop

Class 16

Common Errors in Loop

Re-declaring a variable inside the loop body:

```
int sum=0;
int count=0;

while(count<5)
{       count ++ ;
        // this is replacing the sum declared
        // on the outer scope
        int sum = sum  + count ;
}       // scope of inner sum ends here on each loop iteration

cout<<" " << sum; // output would be 0
```

Sentinel Loop

```
int value = 1;

while(value !=0) // sentinel condition
{       cout<<"\nEnter a value: (0 to exit)";
        cin>>value;
}
```

- The loop continues as long as the sentinel condition is true

- In this example the sentinel condition is value ==0

Interactive Loop

```
int value = 0;
char moredata = 'y';
while(moredata == 'y') // interaction condition
{       cout<<"\nEnter a value: ";
        cin>>value;
        cin.ignore();

        cout<<"\nDo you want more data? [y/n]: ";
        cin>>moredata;
}
```

- On each iteration of the loop, we ask the user whether they want to continue one more time. Depending on user's response the loop would make iteration one more time or stop

The do-while Loop

- the second loop construct of C++ is the do-while loop
- its form is below — note the semicolon!

```
do
{
   statement;
   statement;
   ...
} while (expression);
```

- this is a posttest loop
- its Boolean expression is tested after the loop body executes
- it is guaranteed that the loop body will execute at least once
- look at program count_accumulate_do_while.cpp, which is the previous program converted to use a do-while loop

The do-while Loop

```
int a = 0;
while(a>0)
{
        a = a - 1; }
cout<<a<<endl;
```

```
int a = 0;
do
{       a = a - 1;
}while(a>0);
cout<<a<<endl;
```

Regardless of whether the loop condition is true or false

the do .. while loop will execute at least once

In certain cases, this behavior is advantageous

In the above, for the while loop, the output would be 0 as the pretest is false

In case of do..while loop the loop body was executed and stopped after the postest, the ouput would be -1

Controlling a Loop With a Flag

```
1   bool done = false;
2
3   while (!done)
4   {
5     cout << "Enter a plan: ";
6     char plan;
7     cin >> plan;
8
9     if (plan == 'A')
10       // stuff for plan A ...
11    else if (plan == 'B')
12       // stuff for plan B ...
13    else if (plan == 'C')
14       // stuff for plan C ...
15    else
16    {
17      done = true;
18    }
19  }
```

- it is extremely common to control a while loop with a Boolean flag
- the flag is initialized to false
- when the loop exit condition is recognized, the flag is set to true

Input Validation

- a common use for a while or a do-while loop is input validation

```
unsigned act_score;
bool valid_score;

do
{

  cout << "Please enter an ACT score: ";
  cin >> act_score;
  valid_score = act_score < 1 || act_score > 36;
  if (valid_score == false)
  { cout << "Invalid score. Try again" << endl;
  }


}
while (valid_score == false);

// now act_score is valid, so use it
```