Control Statements in 'C'

Control Statements

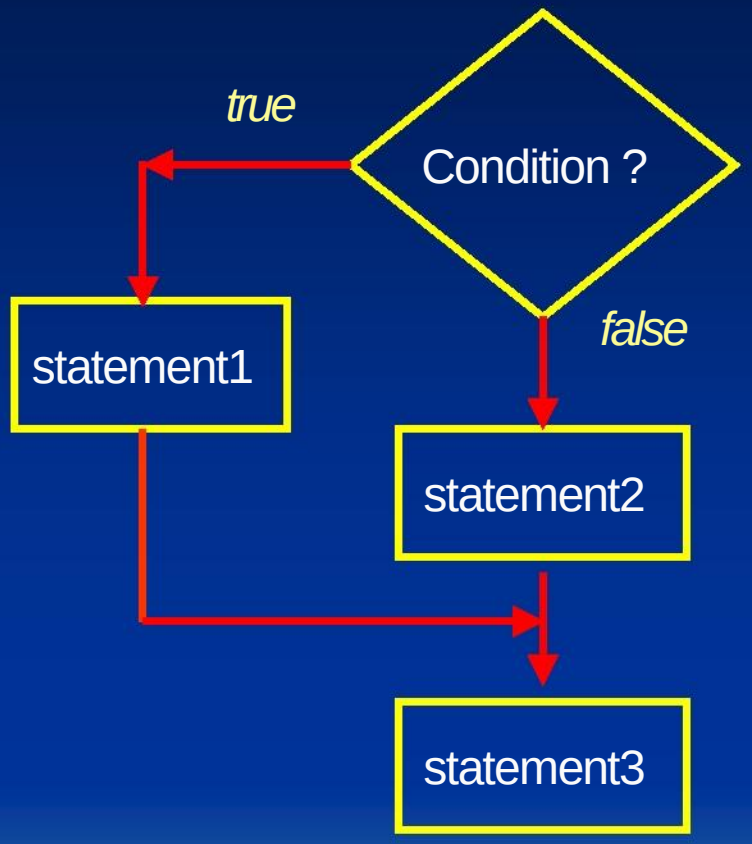
- Selection Statements
  - -Using `if` and `if...else`
  - -Nested `if` Statements
  - -Using `switch` Statements
  - -Conditional Operator
- Repetition Statements
  - -Looping: `while`, `do-while`, and `for`
  - -Nested loops
  - -Using `break` and `continue`

Selection Statements

- `if` Statements
- `switch` Statements
- Conditional Operators

`if` Statements

```
if (Condition)
{
    statement(s);
}
```



Example:

```
if (i > 0)
{
    printf("i = %d ", i );
}
```

Caution

Adding a semicolon at the end of an if clause is a common mistake.

```
if (radius >= 0);
{
    area = radius*radius*PI;
    printf ("The area for the circle
    of radius %d is =%d", radius,area);
}
```

Wrong

This mistake is hard to find, because it is neither a compilation error nor a runtime error, it is a logic error. This error often occurs when you use the next-line block style.

The `if...else` Statement

```
if (condition)
{
   statement(s)-for-the-true-case;
}
else
{
   statement(s)-for-the-false-case;
}
```

`if...else` Example

```
if (radius >= 0)
{
  area = radius*radius*PI;
  printf("The area for the circle of radius
     %d is =%d" , radius, area);
}
else
{
  printf("Radius can not be Negative ");
}
```

Multiple Alternative if Statements

```
if (score >= 90)
   grade = 'A';
else
   if (score >= 80)
      grade = 'B';
   else
      if (score >= 70)
         grade = 'C';
      else
         if (score >= 60)
            grade = 'D';
         else
            grade = 'F';
```

```
if (score >= 90)
   grade = 'A';
else if (score >= 80)
   grade = 'B';
else if (score >= 70)
   grade = 'C';
else if (score >= 60)
   grade = 'D';
else
   grade = 'F';
```

Note

The else clause matches the most recent if clause in the same block. For example, the following statement

```
int i = 1; int j = 2; int k = 3;
if (i > j)
   if (i > k)
   printf("A");
   else
   printf("B");
```

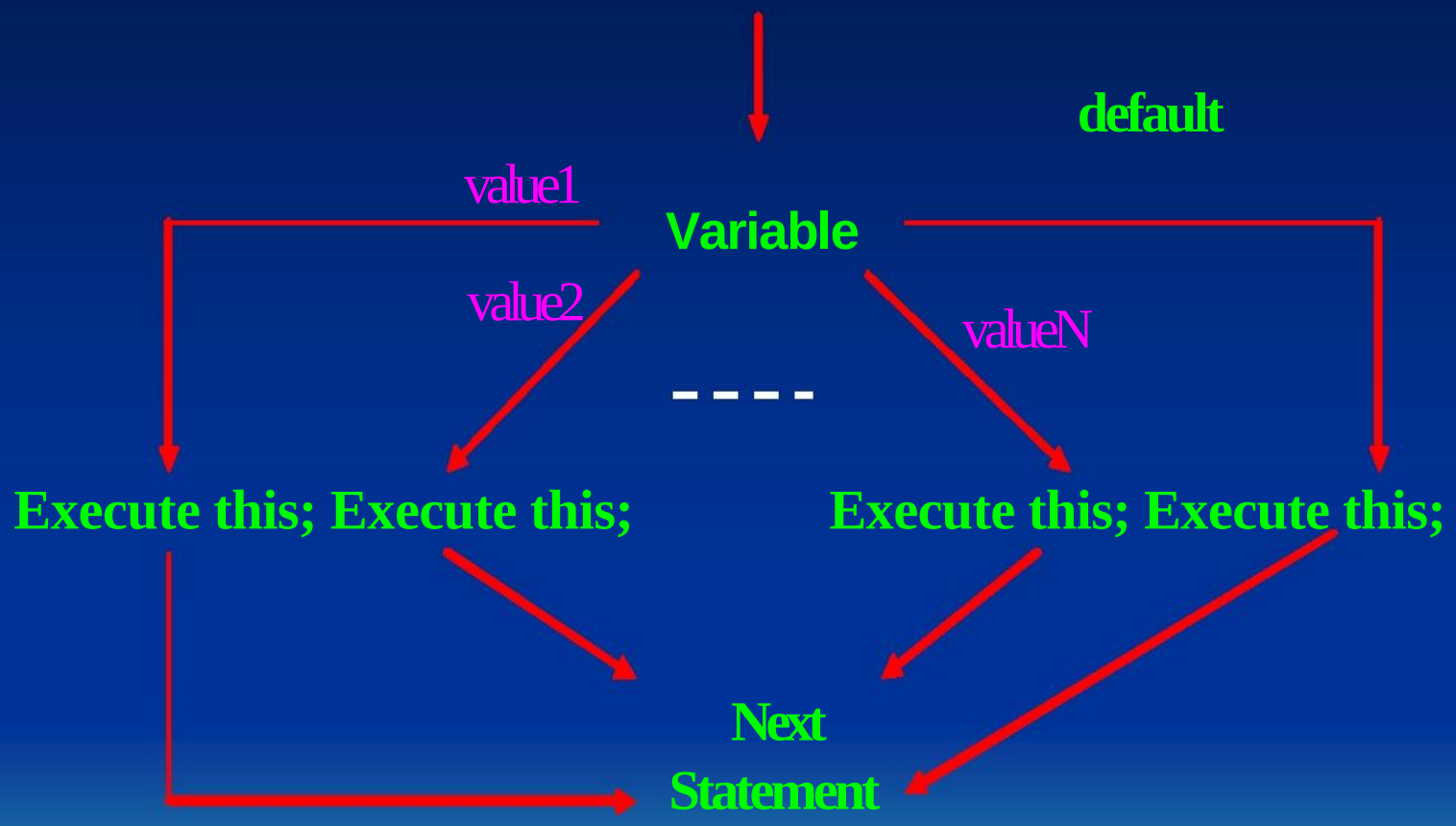
Other combination is :

```
int i = 1; int j = 2; int k = 3;
if (i < j)
{
    if (i > k)
    printf("A");
}
else
    printf("B");
```

switch Statements

```
switch (variable-name)
{
   case value1:
                    Execute this;
                    break;
   case value2:
                    Execute this;
                    break;
          ¦
          ¦
          ¦
   case valueN:
                    Execute this;
                    break;
   default:
                    Execute this;
}
```

`switch` Statement Flow Chart



`switch` Statement Rules

The switch-expression must yield a value of char or int type and must always be enclosed in parentheses. The value1, ..., and valueN must have the same data type as the value of the switch-expression. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. (The case statements are executed in sequential order.)

The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement.

If the break statement is not present, the next case statement will be executed.

`switch` Statement Rules, cont…..

The default case, which is optional, can be used to perform actions when none of the specified cases is true.

The order of the cases (including the default case) does not matter. However, it is a good programming style to follow the logical sequence of the cases and place the default case at the end.

Caution

Do not forget to use a break statement when one is needed. For example, the following code always displays Wrong number of years regardless of what **num** is. Suppose the **num** is 15. The statement **rate = 8.50** is executed, then the statement **rate = 9.0**, and finally the statement, **printf("Wrong number of years").**

```
switch (num) {
      case 7: rate = 7.25;
      case 15: rate = 8.50;
      case 30: rate = 9.0;
      default: printlf("Wrong number of years");
}
```

Example: switch-case

```
int w = 20;
switch(w)
{
  case 10:      printf("First");
                break;
  case 20:      printf("Second");
                break;
  case 30:      printf("Third");
                break;
  default:      printf("Wrong value...");
}
```

Conditional Operator

(condition) ? exp1 : exp2

```
if (x > 0) y = 1
else y = -1;
```

is equivalent to

```
y = (x > 0) ? 1 : -1;
```

Ternary operator

Conditional Operator

```
if (num % 2 == 0)
  printf("%d is even",num);
else
  printf("%d is odd",num);

(num%2==0)?printf("even"):printf("odd");
```

Repetitions

`while` Loops

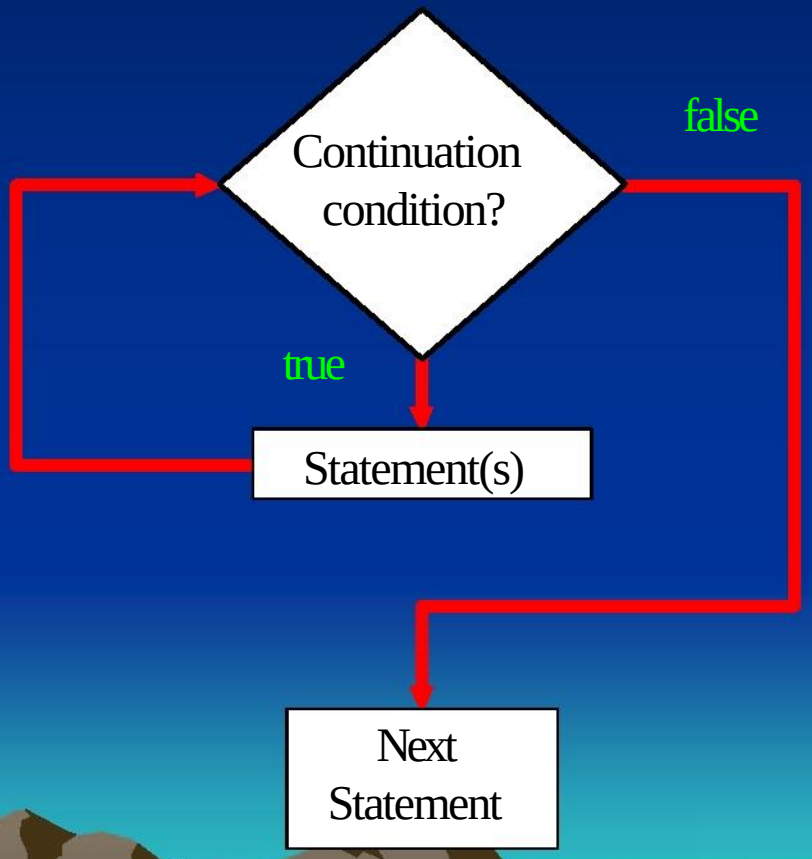
`do-while` Loops

`for` Loops

`break` and `continue`

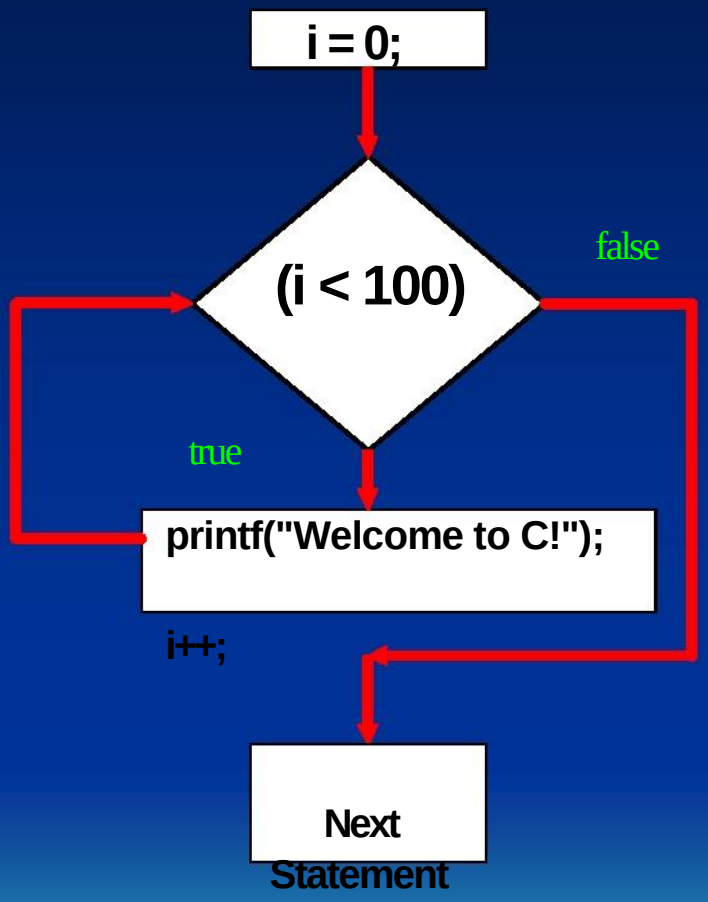
`while` Loop Flow Chart

```
while (continuation-condition)
{
  // loop-body;
}
```

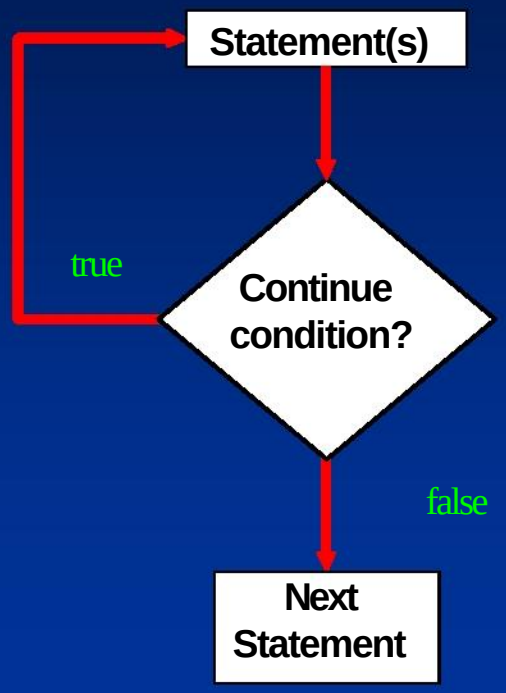


`while` Loop Flow Chart, cont.

```
int i = 0;
while (i < 100)
{
 printf("Welcome to C!");
 i++;
}
```



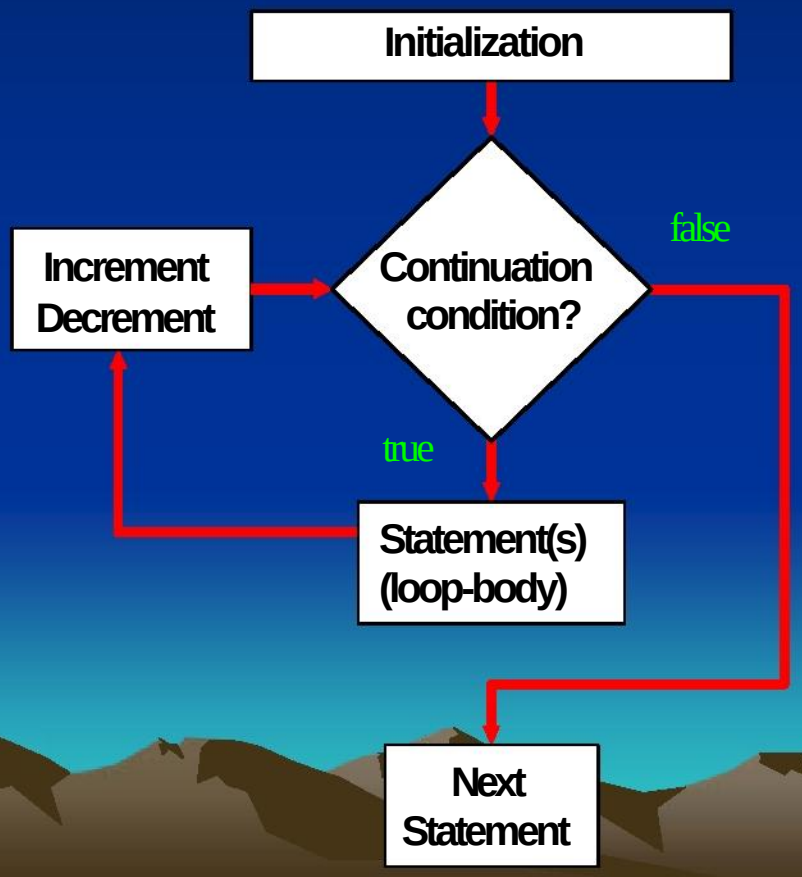
do-while Loop



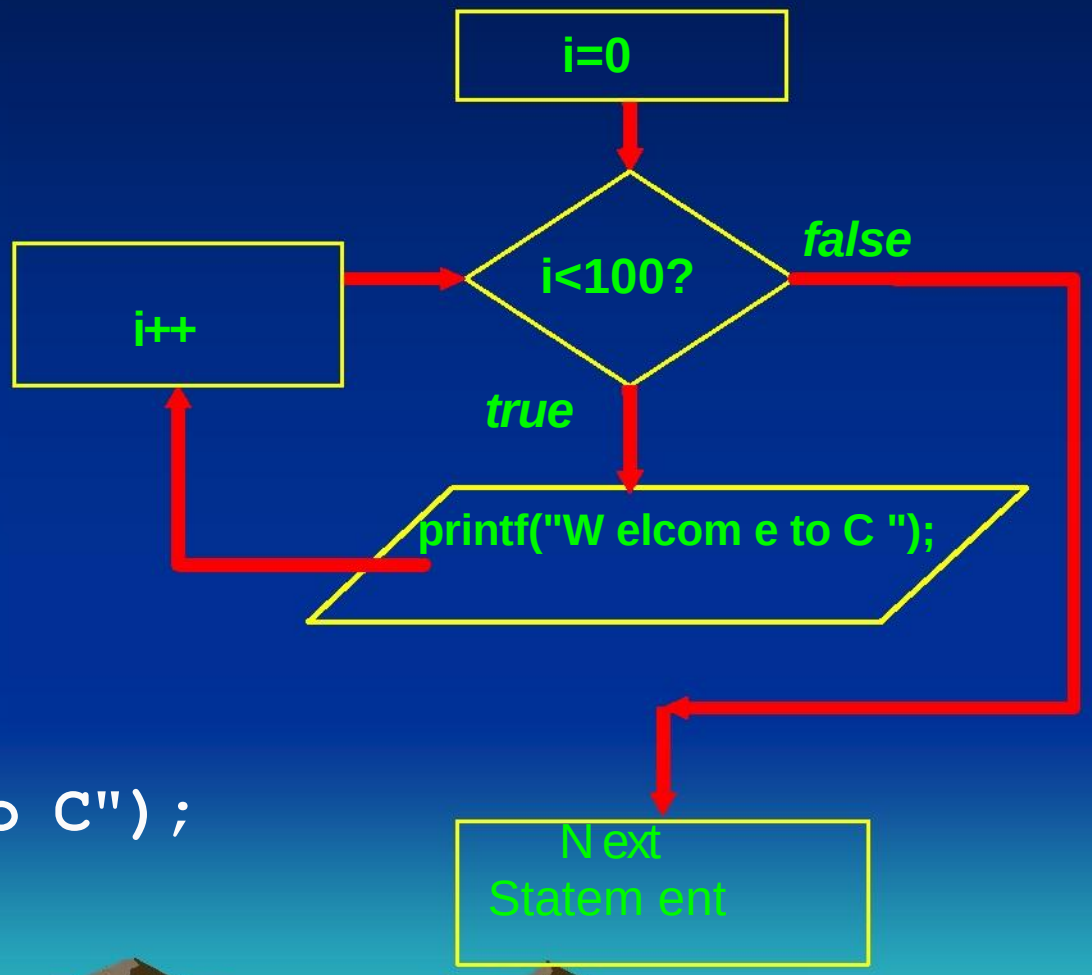

```
do
{
   // Loop body;
} while (continue-condition);
```

`for` Loop

```
for (initialization; condition; increment/decrement)
{
        //loop body;
}
```



`for` Loop Example




```
int i;
for (i=0;i<100;i++)
{
  printf("Welcome to C");
}
```

Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

Wrong

```
for (int i=0; i<10; i++);
{
  printf("i is %d",i);
}
```

Caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;
while (i<10);
{
   printf("i is %d",i);
   i++;
}
```

Wrong

In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do
{
   printf("i is %d",i);
   i++;
} while (i<10);
```

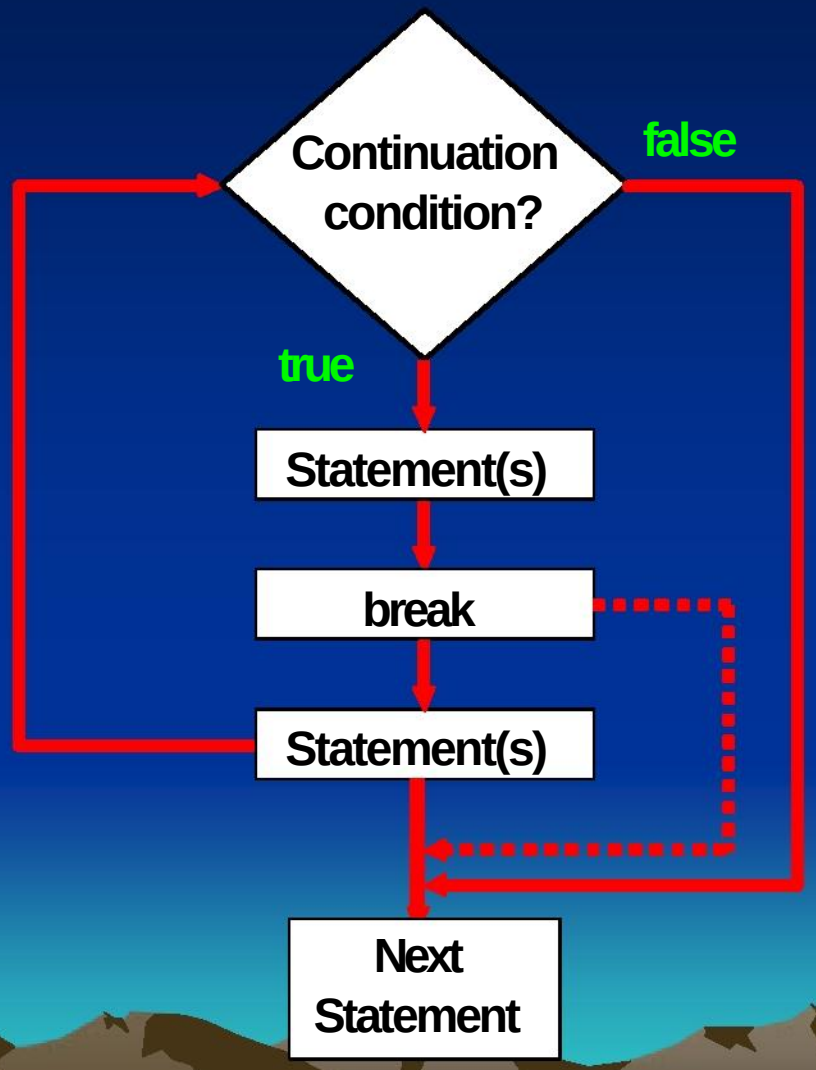
Correct

Which Loop to Use?

The three forms of loop statements, while, do, and for, are expressively equivalent; that is, you can write a loop in any of these three forms.

It is recommend that you use the one that is most intuitive and comfortable for you. In general, a for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times. A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0. A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.

The `break` Keyword



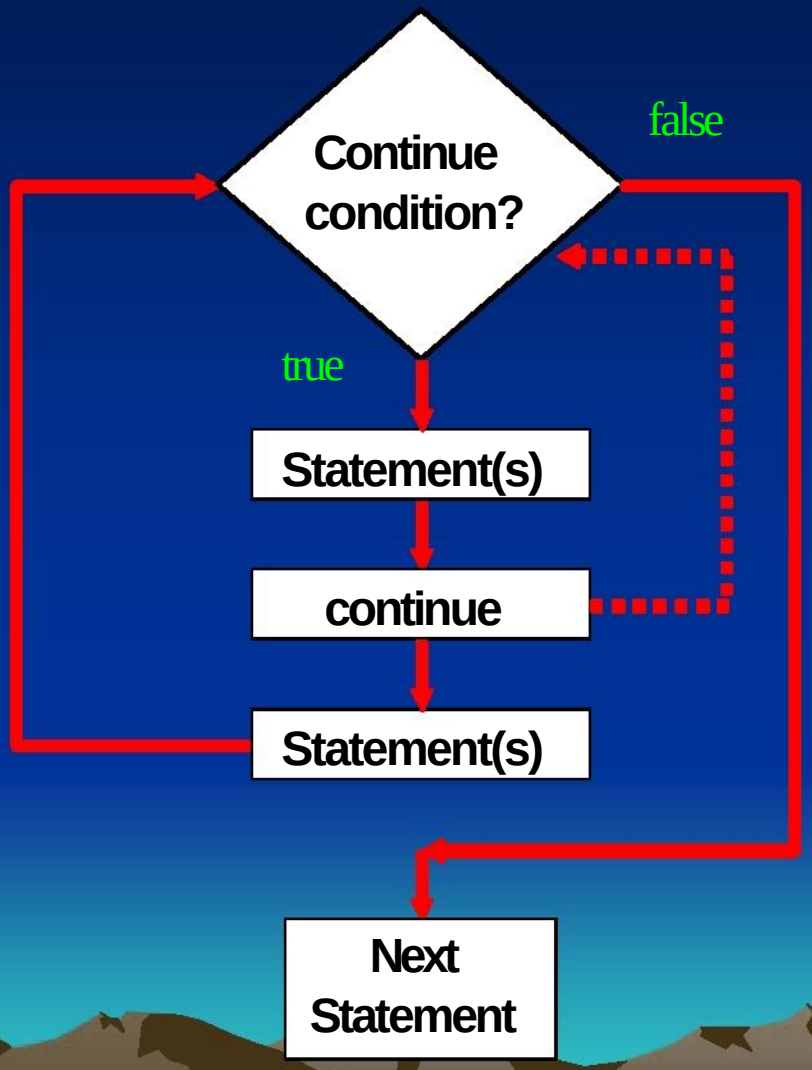
Example: **`break`** statement

```
int a = 10;
while( a >= 0 )
{
        printf("\nValue of a = %d",a);
        a--;
        if(a==5)
        break;
}
```

Output:

```
Value of a = 10
Value of a = 9
Value of a = 8
Value of a = 7
Value of a = 6
```

The `continue` Keyword



Example: `continue` statement

```
int a = 6;
while( a >= 0 )
{
        a--;
        if(a==3)
        continue;
        printf("\nValue of a = %d",a);
}
```

Output:

```
Value of a = 5
Value of a = 4
Value of a = 2
Value of a = 1
Value of a = 0
```

Test your skills…

`If - else` construct

1. Find the largest number from three inputted numbers.

2. Find whether the accepted year is leap year or not ?

Test your skills…

`while loop` construct

1. Display all the even numbers from 20 to 300.

2. Find whether the entered number is prime or not ?

3. Calculate squares of all the numbers from -10 to 50.

Test your skills…

`for loop` construct

1. Calculate the factorial of the given number n.

2. Calculate sum of digits of given number.

3. Calculate squares of all the numbers from -10 to 50.