Programming Fundamentals

BE(EE)-3

Chapter No 03
Control Structure
Loop & Decisions

Course Covered

- Relational Operators
- Loops
- Decisions
- Logical Operators
- Precedence Summary
- Other Control Statements

Relational and Equality Operators

- Relational and Equality operators are used to compare values:
- Relational Operators:

>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal

- Equality Operators:

==	Equal to
!=	Not Equal to

Relational and Equality Operator

- The relational operators have very low precedence and associate left-to-right.
- The equality operators have very-very low precedence and associate left-to-right.
- Some examples:

```
17 < x          var == 3.14
age != 21       x+1 >= 4*y-z
```

Assignment Operator

- The assignment operator "=" is used to assign a value to a variable.

```
x = 13 - y;
```

- Assignment has very low precedence and associates from right to left.
- You can do this:

```
x = y = z + 15;
```

Precedence

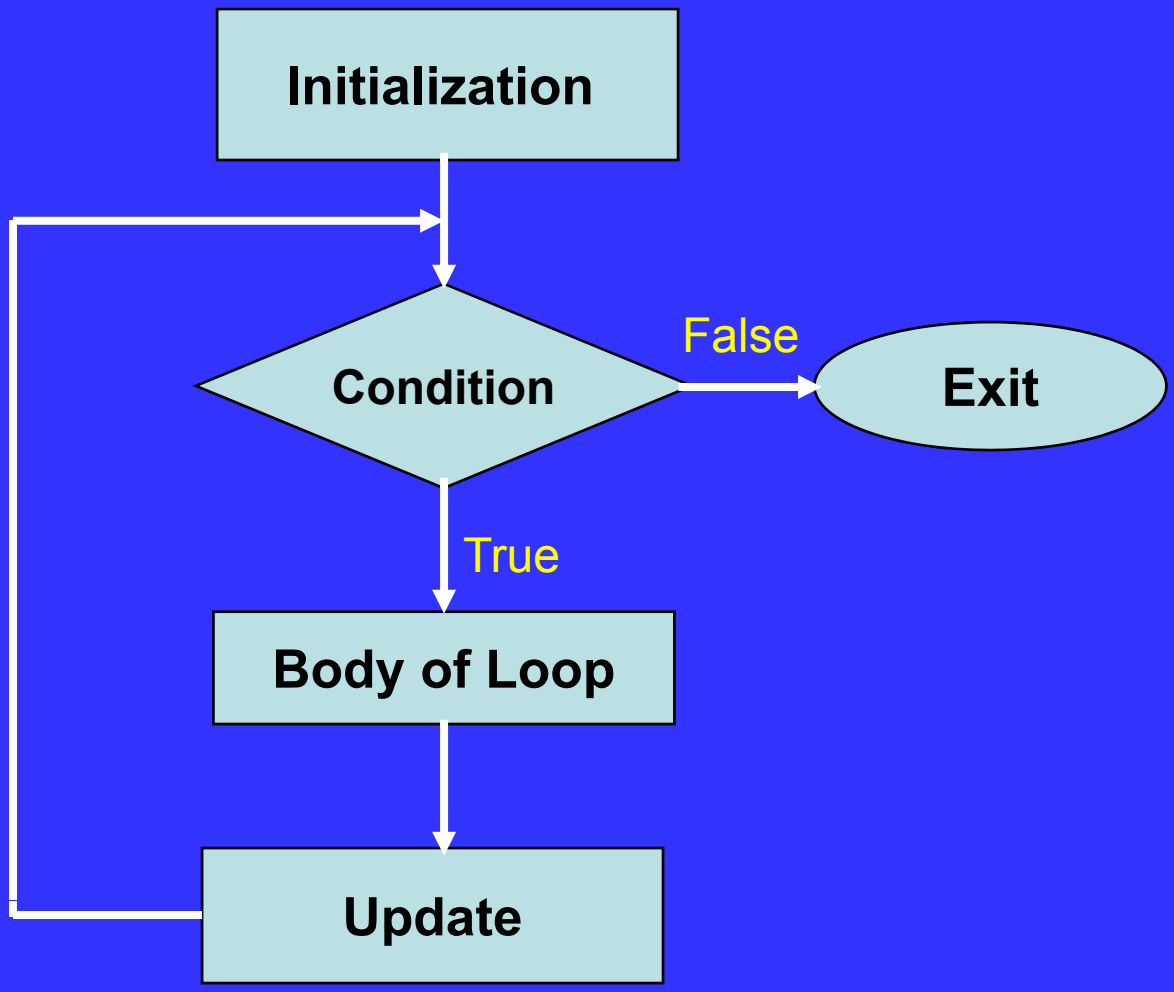
Operators	Precedence
( )	highest (applied first)
* / %	
+ -	
< <= > >=	
== !=	
=	lowest (applied last)

LOOPS : Repetition Structures

A control structure allowing a program to do something either definite or indefinite number of times.

- The For Loop
- The While Loop
- The do Loop

Operation of `for` Loop



Conditions

- The *condition* used in a control structures is a Boolean value - either `true` or `false`.
- In C++:
  - the value 0 is `false`
  - anything else is `true`

`for` loops

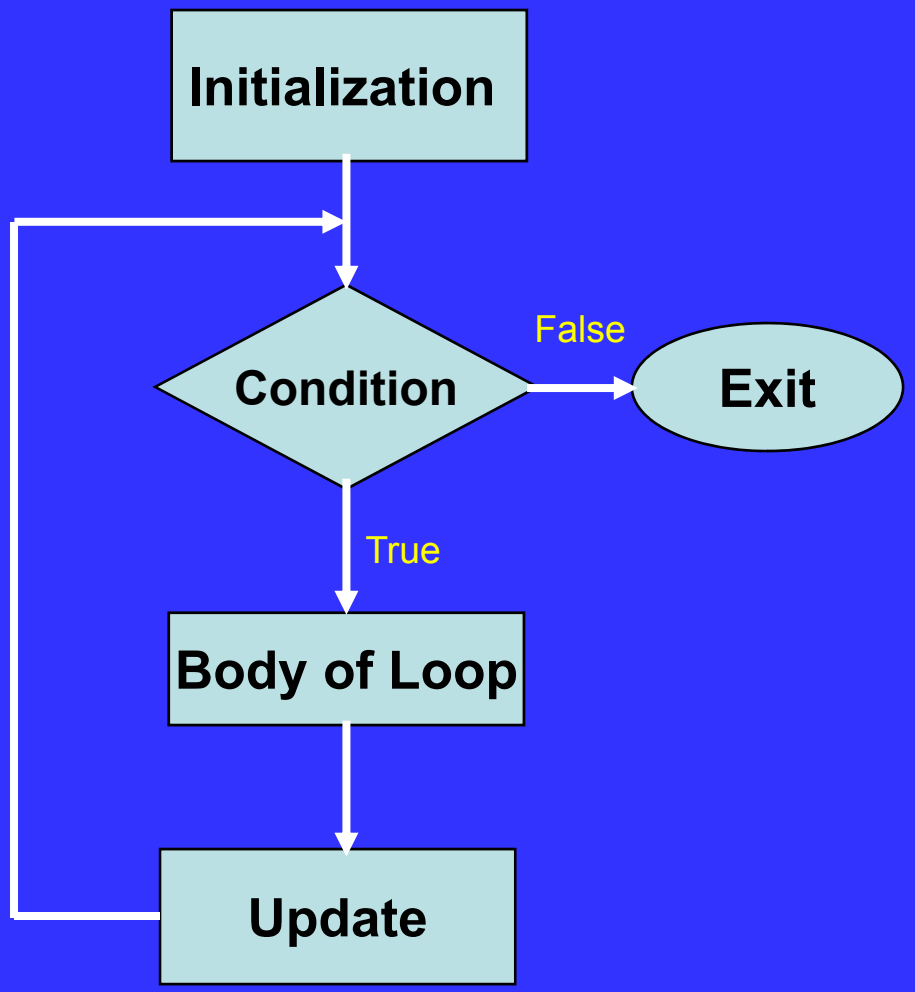
- The `for` control structure is often used for loops that involve doing something for a fixed number of times.

Syntax Style

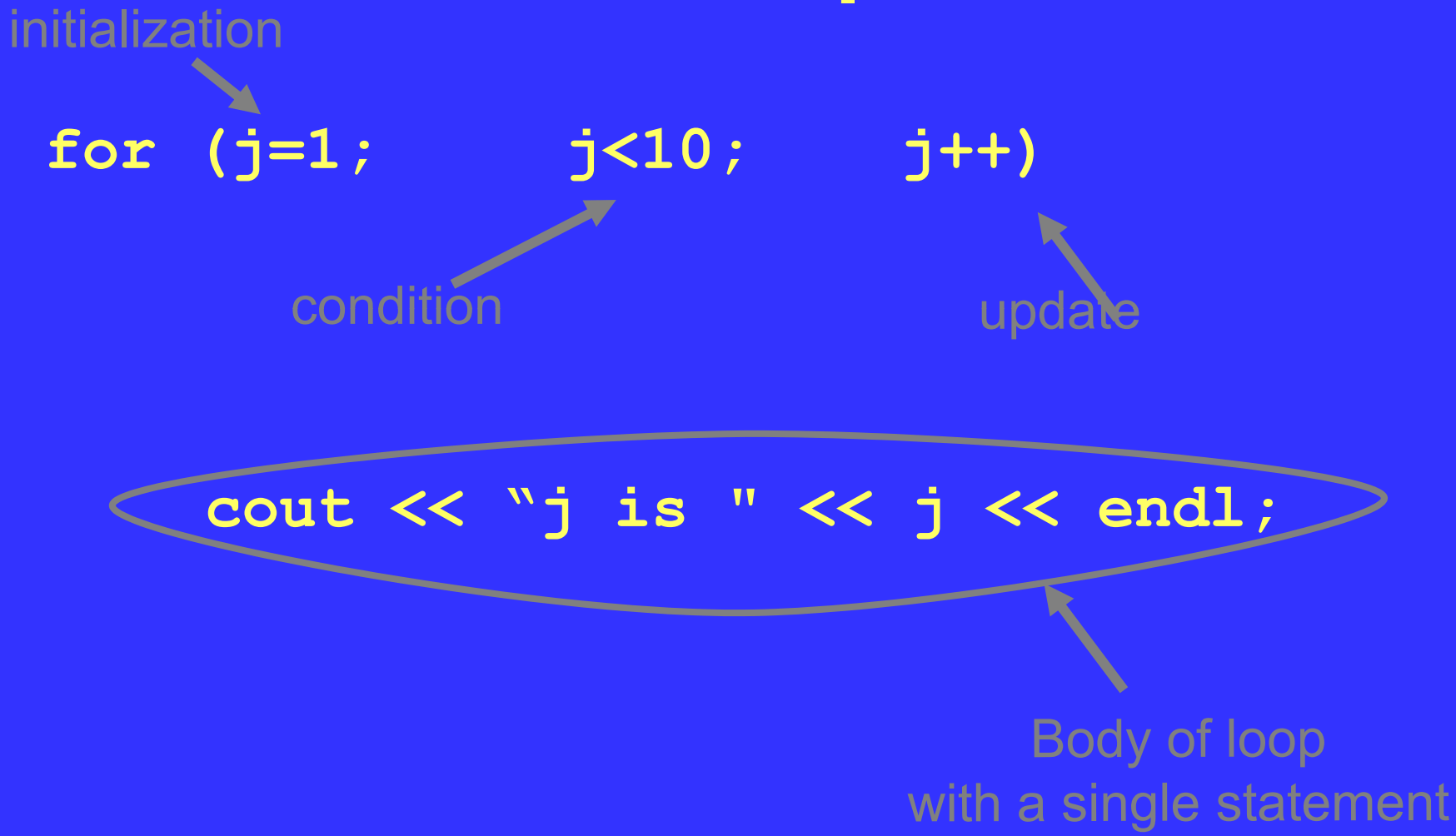
```
for (initialization; condition; update)
   Do something;
```

```
for (initialization; condition; update)
```

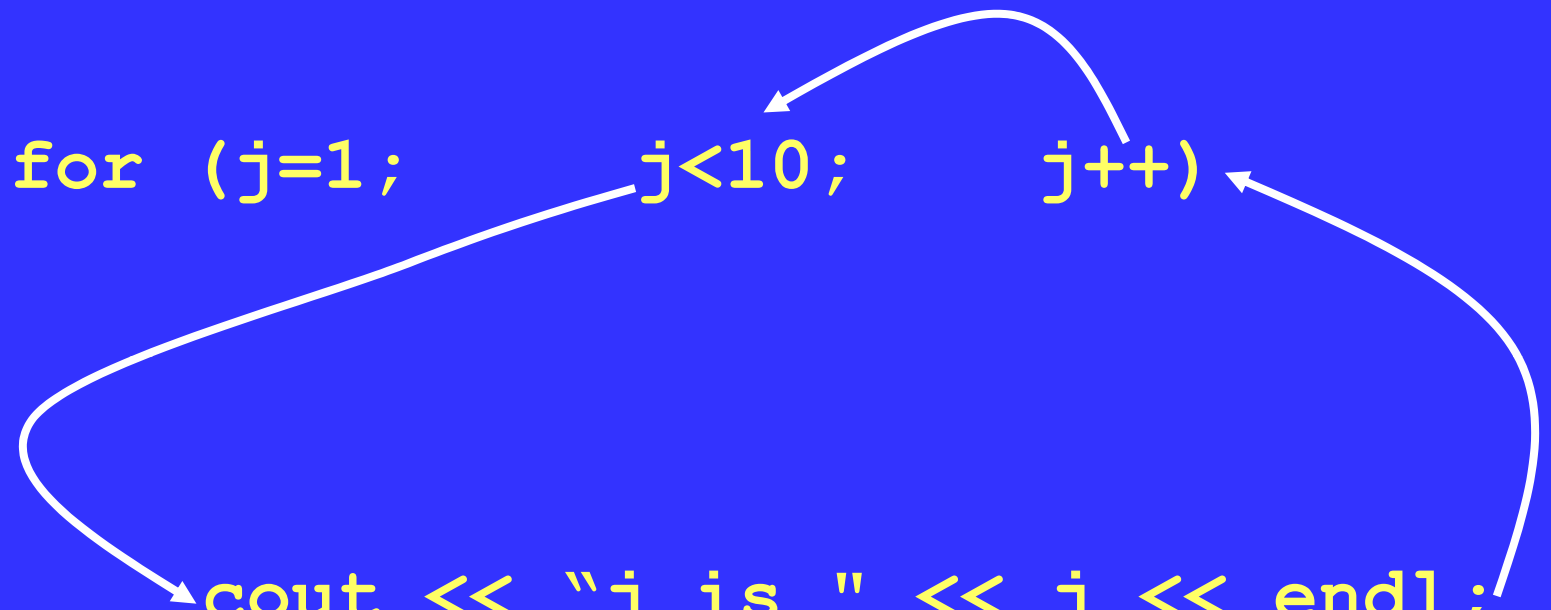
- initialization is a statement that is executed at the beginning of the loop (and never again).
- the body of the loop is executed as long as the condition is true.
- the update statement is executed each time the body of the loop has been executed (and before the condition is checked)



`for` example


initialization
for (j=1; j<10; j++)
condition
update
cout << "j is " << j << endl;
Body of loop
with a single statement

For Loop



How Many Times?

```
for (j=1; j<10; j++)      9 times
for (j=1; j<=10; j++)    10 times
for (j=0; j<10; j++)     10 times
for (j=0; j<=10; j++)    11 times
for (j=10; j>1; j--)      9 times
for (j=10; j>=1; j--)    10 times
for (j=10; j>0; j--)     10 times
for (j=10; j>=0; j--)    11 times
```

`for` example

```
// fordemo.cpp
//demonstrates simple for loop
#include <iostream.h>

    void main()
    {
    int j;                    //define a loop variable
    for(j=0; j<15; j++){          //loop from 0 to 14,
    cout<<j*j<< " " ;    // displaying the square of j
    cout << endl;
    }
    }
```

How many statements are there in loop body?

MULTIPLE STATEMENTS IN LOOP

```
// cubelist.cpp
// lists cubes from 1 to 10
#include <iostream.h>
#include <iomanip.h>     //for setw

int main()
    {
    int numb;  //define loop variable
    for (numb=1; numb<=10; numb++) //loop from 1 to 10
        {
            cout << setw(4) << numb; //display 1st column
            int cube = numb*numb*numb; //calculate cube
            cout << setw(6) << cube << end1; //display 2nd column
        }
    return 0;
    }
```

Variable Visibility:

Variable defined inside a block is not visible outside it

Statement Block or
Compound Statement

MULTIPLE STATEMENTS IN `for` LOOP

```
// cubelist.cpp
// lists cubes from 1 to 10
#include <iostream.h>
#include <iomanip.h>     //for setw

int main()
    {
    int numb;  //define loop variable
    for (numb=1; numb<=10; numb++) //loop from 1 to 10
          {
                cout << setw(4) << numb; //display 1st column
                int cube = numb*numb*numb; //calculate cube
                cout << setw(6) << cube << end1; //display 2nd column
          }
    return 0;
    }
```

Indentation:

As a good practice loop body (single or compound statement) and delimiting braces if any are shifted relatively to the right.

For Loop Variations

```
// factor.cpp
// calculates factorial, demonstrates for loop
#include <iostream.h>


int main()
{
unsigned int numb;
unsigned long fact=1;
cout<<"Enter a number:  ";
cin>>numb;

for(int j=numb; j>0; j--)
    fact *= j;

cout<<"Factorial of "<<numb<<" is "<<fact<<endl;
return 0;
}
```

More about `for`

- You can leave the initialization, condition or update statements blank.
- If the condition is blank the loop never ends!

```
for (i=0; ;i++)
   cout << i << endl;
```

And What if we have

```
for (;;)
```

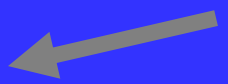
Unsigned Data Types

```
// signtest.cpp
// tests signed and unsigned integers
#include <iostream.h>


int main()
   {
   int signedVar = 1500000000;          //signed
   unsigned int unsignVar = 1500000000;  //unsigned

   signedVar = (signedVar * 2) / 3;  //calculation exceeds range
   unsignVar = (unsignVar * 2) / 3;  //calculation within range

   cout << "signedVar = " << signedVar << endl;  //wrong
   cout << "unsignVar = " << unsignVar << endl;  //OK
   return 0;
   }
```

The `while` Loop

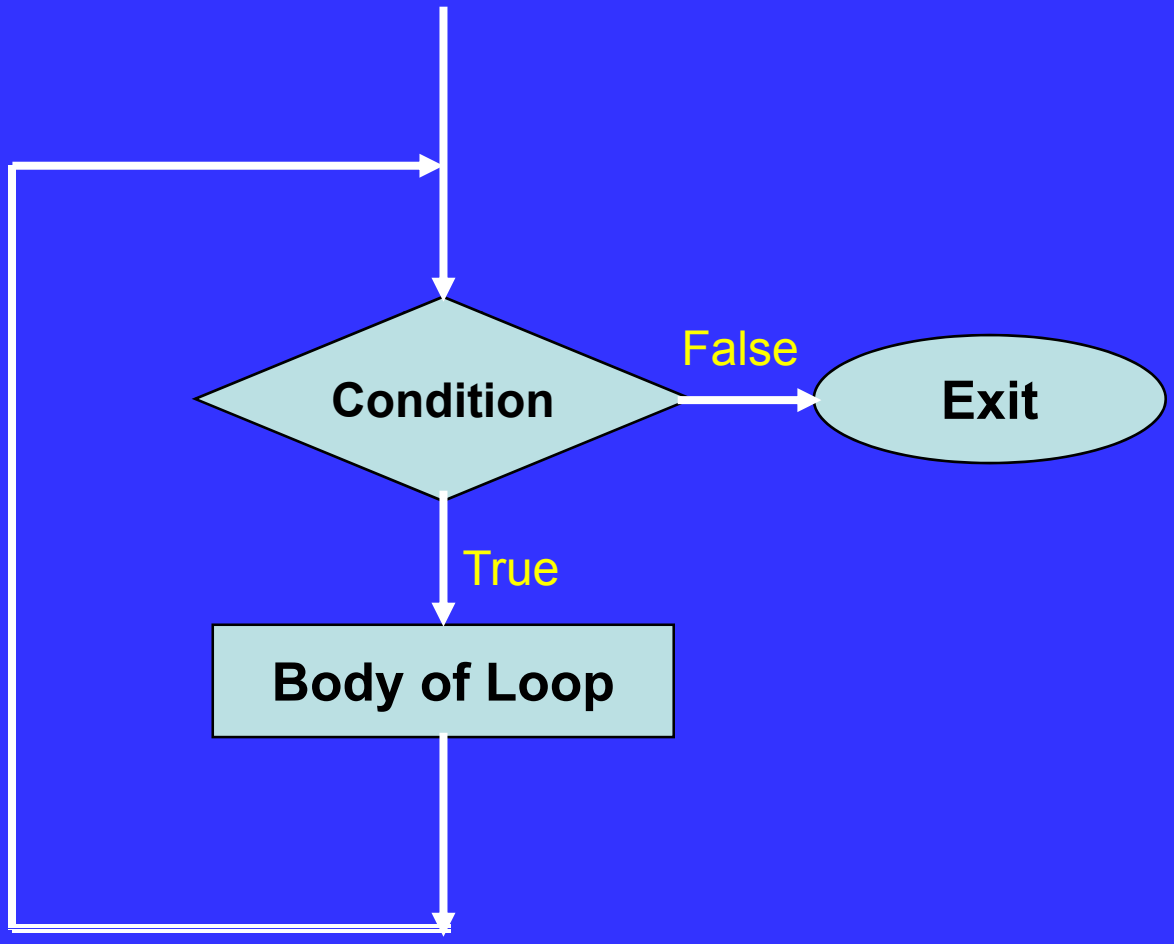
- The `while` control structure supports repetition - of a statement (or compound statement) until the condition is false.
- The test of the termination condition takes place before each execution of the loop
- Syntax

```
while (expression )
{
    // statements;
}
```



body of the loop

Operation of `while` Loop



Example

```
// endon0.cpp
// demonstrates WHILE LOOP
#include<iostream.h>

void main()
   {
   int n = 99;          // make sure n isn't initialized to 0
   while( n !=0)        // loop until n is 0
   cin >> n;            //read a number into n
   cout << endl;
   return 0;
   }
```

```
1   27    33        144       9         0
```

MULTIPLE STATEMENTS IN `while` LOOP

```
// while4.cpp
// prints numbers raised to fourth power

#include<iostream.h>
#include<iomanip.h>
int main()
{
     int pow=1, numb=1;

     while(pow<10000)
     {
          cout<<pow;
          pow =pow *8;
     }
cout<<endl;
return 0;
}
```

Example Precedence `while` LOOP

```
// fibo.cpp
// demonstrates higher precedence of arithmetic operators
#include<iostream.h>

int main()
{
    const unsigned long limit = 4294967295;
    unsigned long next=0, last=1;

    while (next<limit/2)
    {
        cout<<last<<"    ";
        long sum=next +last;
        next=last;
        last=sum;
    }

    cout<<endl;
    return 0;
}
```

1 1 2 3 5 8 13 21 34 55

……..

……..

Type Name	Bytes	Other Names	Range of Values
int	4	signed	-2,147,483,648 to 2,147,483,647
unsigned int	4	unsigned	0 to 4,294,967,295

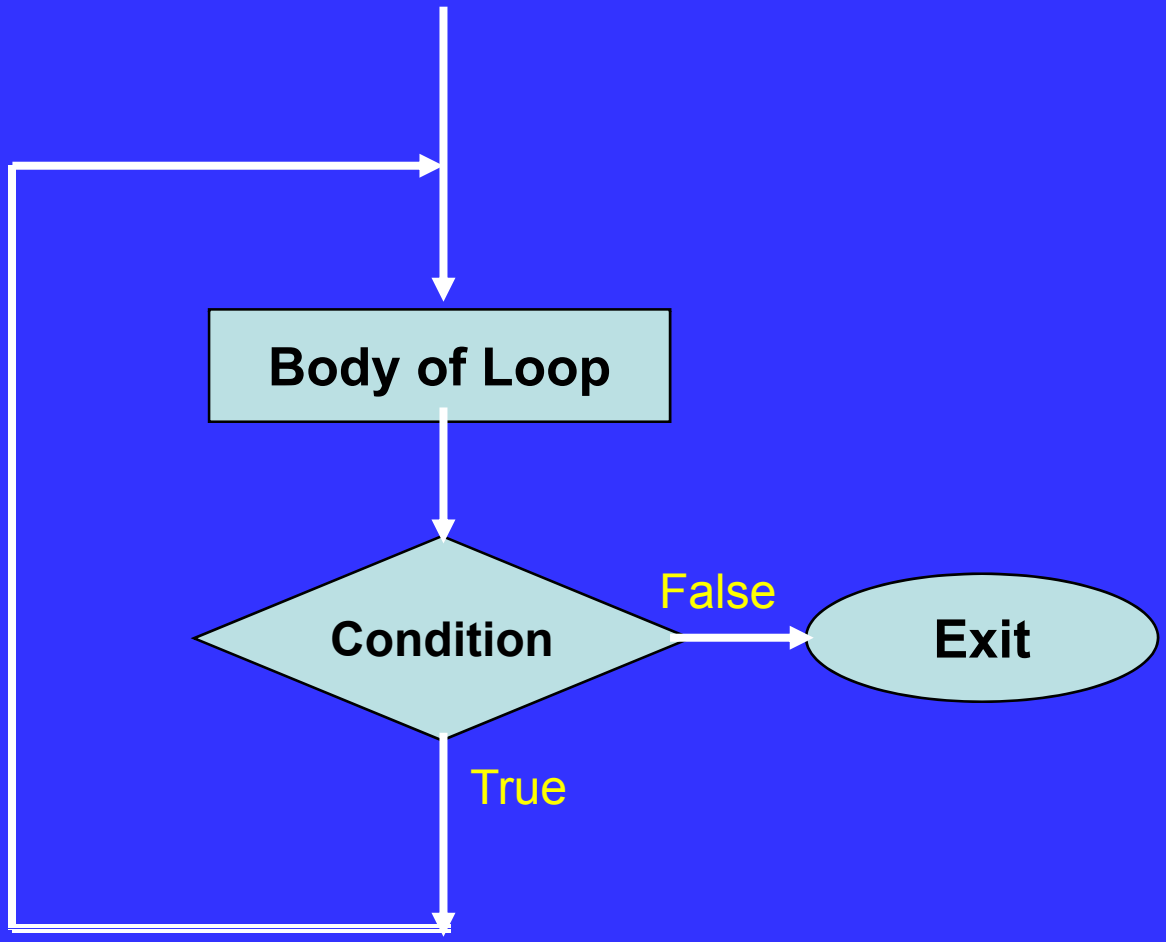
The do Loop

- The `do while` control structure also provides repetition, this time the condition is at the bottom of the loop.
  - the body is executed at least once

- Syntax

```
do
{
  // statements;
} while (expression);
```

Operation of `do` Loop



Example

```
// divdo.cpp
// demonstrates DO loop
#include <iostream.h>

int main()
    {
    long dividend, divisor;
    char ch;

    do                                          //start of do loop
        {                                       //do some processing
        cout << "Enter dividend: "; cin >> dividend;
        cout << "Enter divisor: "; cin >> divisor;
        cout << "Quotient is  " << dividend / divisor;
        cout << " , remainder is " << dividend % divisor;
        cout << "\nDo another? (y/n): "; //do it again?
        cin >> ch;
        }
    while( ch != 'n' );  //loop condition
    return 0;
    }
```

DECISION

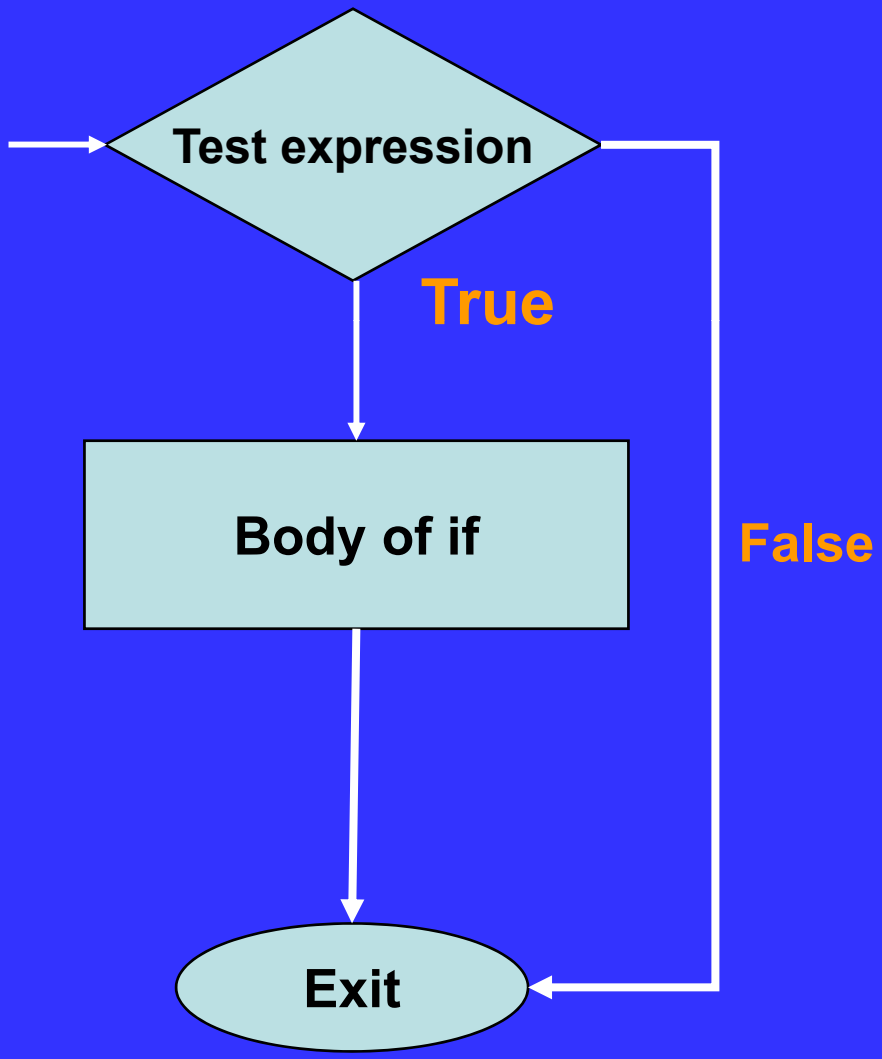
- The if Statements
- The if … else Statement
- Nested if ….. else Statement
- The switch/break Statement
- The continue Statement
- The goto Statement

`if` Statement

- The `if` control structure allows us to state that an action (sequence of statements) should happen only when some condition is true:

```
if (condition)
    action;
```

Operation of if Statement



`if` example

```
// ifdemo.cpp
// demonstrates IF statement
#include <iostream.h>

int main()
    {
    int x;
    cout << "Enter a number:  ";
    cin >> x;
          if( x > 100)
          cout << "That number is greater than 100\n";
    return 0;
    }
```

Multiple statements in `if`

```
// if2.cpp
// demonstrates IF statement
#include <iostream.h>

int main()
    {
    int x;
    cout << "Enter a number:  ";
    cin >> x;
          if( x > 100)
          {
                cout << "The number "<<x;
                cout << " is greater than 100\n";
          }
    return 0;
    }
```

More ifs

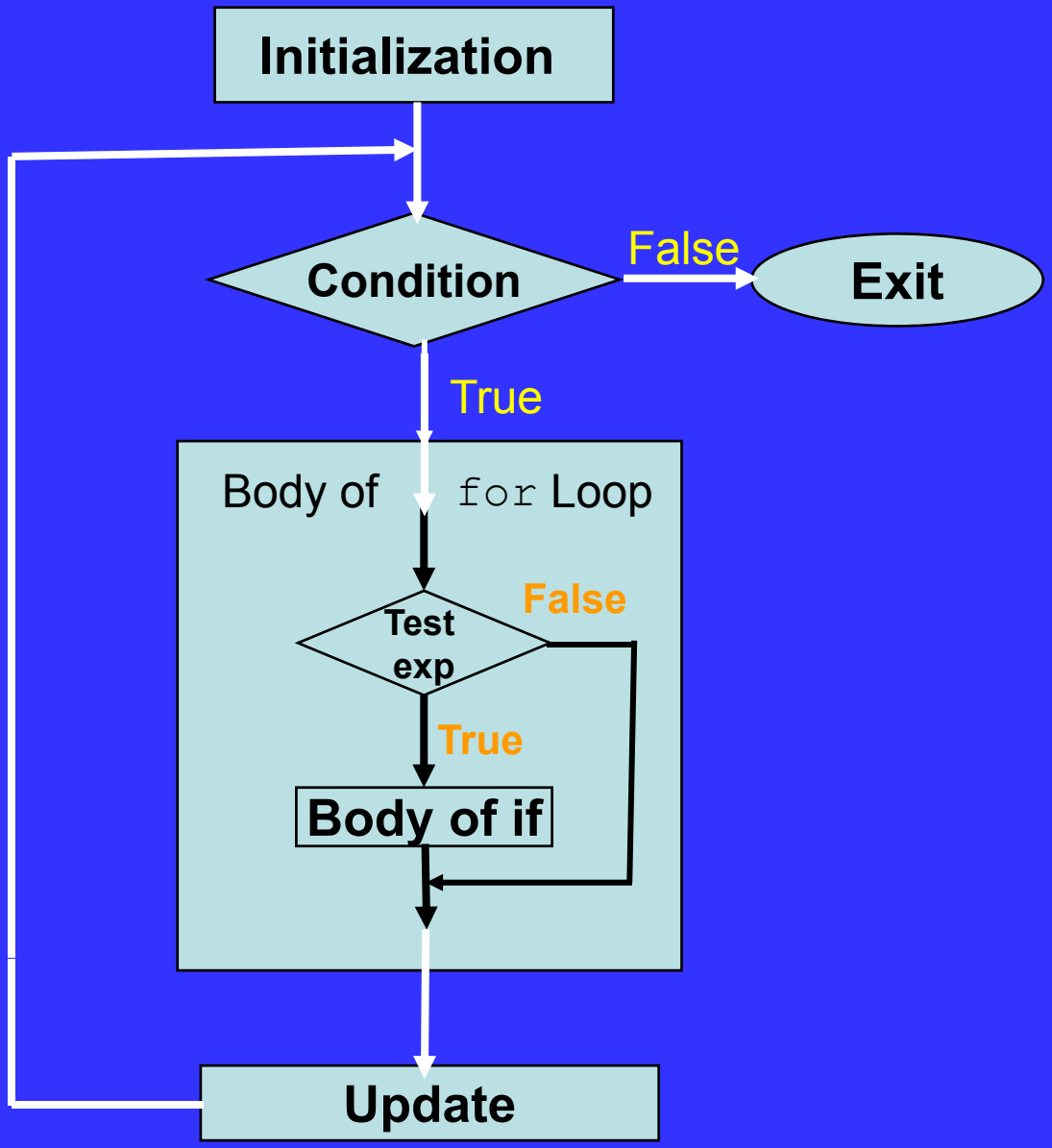
```
if (var)
   cout << “var is not zero\n";

if (marks>=90)
  grade = 'A';

if (grade == 'F')
  cout << “You have failed!!\n"
```

Common Mistake

- It is easy to mix up the assignment operator "=" with the equality operator "==".
- What's wrong with this:

```
if (marks=100)
   cout << "your grade is perfect!\n";
```

Nesting `if` inside a `for` Loop



Example

```
// prime.cpp
// Demostrates a nested if inside a for loop

#include<iostream.h>
#include<process.h>


int main()
{
     unsigned long n, j;
     cout<<"Enter a number: ";
     cin>>n;
     for(j=2; j<=n/2; j++)
          if(n%j == 0)
          {
               cout<<"It's not aprime; divisible by "<<j<<endl;
               exit(0);
          }
     cout<<"It's prime\n";
     return 0;
}
```

Library Function `exit()`

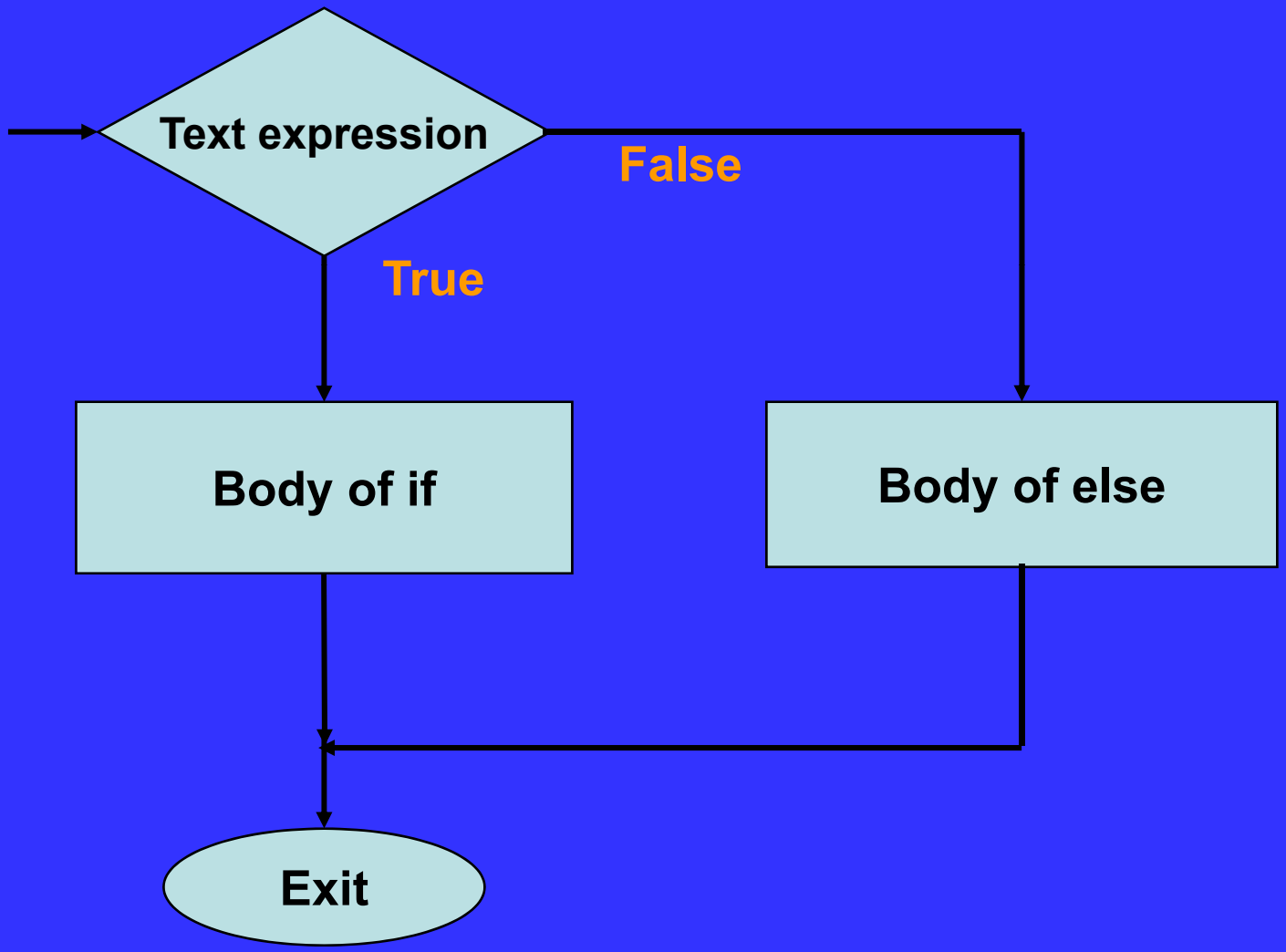
- The function exit(0) causes the program to terminate no matter where it is in the listing

`if else` Control Structure

- The `if else` control structure allows you to specify an alternative action:

```
if ( condition )
     action if true;
else
     action if false;
```

If...else Structure



Example

```
// ifelse.cpp
// demonstrates If… else statement
#include <iostream.h>

int main()
    {
    int x;
    cout << "\nEnter a number:";
    cin >> x;
          if ( x > 100)
          cout << "That number is greater than 100\n";
          else
          cout  << "That number is not greater than 100\n";
return 0;
}
```

`if else` example

```
if (marks >= 90)
  grade = 'A';
else
  grade = 'F';
```

The `getche` Library Function

- The `getche` returns each character as soon as it is typed. This eliminates requirement of pressing “Enter” by the user.
- The ‘e’ in the end of function name indicates that the function echoes the character to the screen.

If...else Embedded in while Loop

```
// chcount.cpp
// counts characters and words typed in
#include<iostream.h>
#include<conio.h>    // for getche


int main()
{
      int chcount=0, wdcount=1;
      char ch = 'a';
      cout<<"Enter a phrase:  ";

      while (ch != '\r')
      {
            ch = getche();
            if(ch==' ')
                  wdcount++;
            else
                  chcount++;
      }
      cout<<"\nwords="<<wdcount<<endl<<"Letters="<<(chcount-1)<<endl;
      return 0;
}
```

Another Way of Writing `while`

```
// chcnt2.cpp
// counts characters and words typed in
#include<iostream.h>
#include<conio.h>    // for getche


int main()
{
      int chcount=0, wdcount=1;
      char ch = 'a';
      cout<<"Enter a phrase:  ";

      while ((ch = getche())!= '\r')
      {
            if(ch==' ')
                  wdcount++;
            else
                  chcount++;
      }
      cout<<"\nwords="<<wdcount<<endl<<"Letters="<<(chcount-1)<<endl;
      return 0;
}
```

Nested if…else Statement

- A nested `if` is the target of another `if` or `else`.

```
if(i)
{
  if(j) statement 1;
  if (k) statement 2;   // this if
  else statement 3;    // is associated with this else
}
 else statement 4 ;  // associated with if (i)
```

- `else` refers to the nearest `if` that is within the same block.
- An `else` is matched with the last `if` that does not have its own `else`

Nested if…else Statement

```
// badelse.cpp
// demonstrates ELSE matched with wrong IF

#include<iostream.h>

int main()
{
    int a, b, c;
    cout<<"Enter three numbers, a, b, and c:\n";
    cin>>a>>b>>c;
    if(a==b)
        if(b==c)
            cout<<"a, b and c are the same\n";
    else
        cout<<"a and b are different\n";
    return 0;
}
```

Case 1: a, b & c same	Only first cout executed
Case 2: a & b different, b & c same	None of the cout executed
Case 3: a & b same, b & c different	Only second cout executed
Case 4: a , b & c all different	None of the cout executed

Nested if…else Statement

```
// correctelse.cpp
// demonstrates ELSE matched with wrong IF

#include<iostream.h>

int main()
{
    int a, b, c;
    cout<<"Enter three numbers, a, b, and c:\n";
    cin>>a>>b>>c;
    if(a==b)
        if(b==c)
            cout<<"a, b and c are the same\n";
        else
        cout<<"b and c are different\n";
    return 0;
}
```

Case 1: a, b & c same	Only first cout executed
Case 2: a & b different, b & c same	None of the cout executed
Case 3: a & b same, b & c different	Only second cout executed
Case 4: a , b & c all different	None of the cout executed

Nested if…else Statement

```
// adifelse.cpp
// demonstrates If…else  with adventure program
#include <iostream.h>
#include <conio.h>      //for getche()


int main()
{
    char dir='a';
    int x=10, y=10;
    cout << "Press Enter to quit\n";
```

Contd.

```
while (dir != '\r')                    // until enter is pressed
    {
    cout << "\nYour location is " << x << ", "<< y;
    cout << "\nPress direction key (n, s, e, w): " ;
    dir = getche();                              // get character
    if ( dir =='n')                    // go north
          y--;
    else
          if ( dir =='s')              // go south
                y++;
          else
                if ( dir =='e')                // go east
                      x++;
                else
                      if ( dir =='w')                  // go west
                            x--;
    }      // end while
          return 0;
}
```

Nested `else...if` Construction

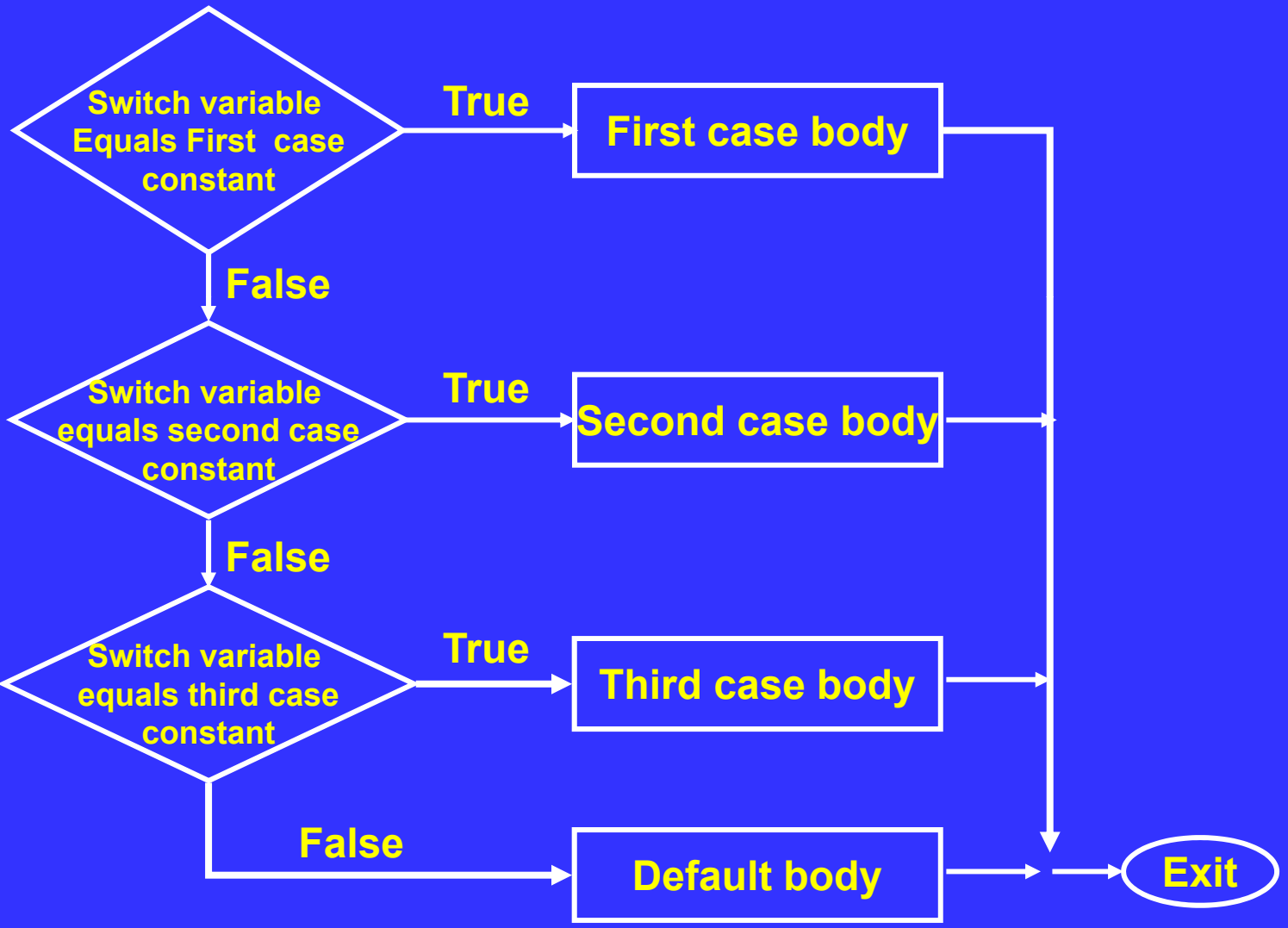
```
// adelseif.cpp
// demonstrates ELSE. If with adventure program
#include <iostream.h>
#include <conio.h>          //for getche()


int main()
{
    char dir='a';
    int x=10, y=10;
    cout << "Press Enter to quit\n";
```

Contd.

Nested `else...if` Construction

```
while (dir != '\r')                    // until enter is pressed
{
cout << "\nYour location is " << x << ", "<< y;
cout << "\nPress direction key (n, s, e, w): " ;
dir = getche();                                 // get character


if ( dir =='n')                        // go north
        y--;
        else if   ( dir =='s')                  // go south
        y++;
        else if   ( dir =='e')                  // go east
        x++;
        else if   ( dir =='w')                  // go west
        x--;
   }  // end while
return 0;
}                                               // end main
```

The switch/break



The switch Statement

Integer or character variable

```
Switch   (n)  ----- Note : no semicolon here
```

Integer or character variable

```
    Case   1: -------- Note : colon here
       statement;
       statement;
       break ;
    Case   2:
       statement;
       statement;
       break ;
    default : -------- Note : colon here
       statement;
       statement;

}  ----- Note : no semicolon here
```

The Break Statement

- The break keyword causes the entire switch statement to exit.
- Control goes to the first statement following the end of the switch construction.

The switch/break Example

```
// platters.cpp
//demonstrates SWITCH statement
#include <iostream.h>

int main()
{
      int speed;                    //turntable speed
      cout << "\nEnter 33, 45, or 78:  ";
      cin >> speed;   //user enters speed
      switch (speed) //selection based on speed
          {
            case 33:   //user entered 33
                cout << "LP album\n";
                break;
            case 45:  //user entered 45
                cout << "Single selection\n";
                break;
            case 78:  //user entered 78
                cout << "Obsolete format\n";
          }
      return 0;
}
```

Another switch/break : Example

```
// adswitch.cpp
// demostratotes SWITCH with adventure program
# include <iostream.h>

#include <conio.h>                                          // for getche( )
int main( )
{
      char dir = 'a' ;
      int x=10, y=10;
      while (dir !='\r' )
            {
            cout << "\nYour location is " << x <<", "<< y;
            cout << "\nEnter direction (n, s, e, w): " ;
            dir = getche();                                 // get character
        switch (dir)                                        // switch on it
                        {
                        case 'n': y--; break;               //go north
                        case 's': y++; break;               //go south
                        case 'e': x++; break;               //go east
                        case 'w': x--; break;               //go west
                        case '\r' : cout << "Exiting\n"; break;   // Enter key
                        default: cout << "try again\n";      // unknown char
                        } // end switch
                }// end while
return 0;
}  // end main
```

Switch Verses if…else

When do you use series of if… else (or else if) statement, and when do you use a switch statement? In an else if construction you can use a series of expressions that involve unrelated variables and are as complex as you like. For example:

```
If( SteamPressure*Factor > 56)
   // statements
Else if( VolageIn + VoltageOut < 23000)
   // statements
Else if( day==Thursday)
  // statements
Else
// statements
```

Contd.

Switch Versus if…else

In Switch statement, however, all the branches are selected by the same variable; the only thing distinguishing one branch from another is the value of this variable. You can't say

```
Case a<3:
// do something
Break;
```

The Conditional Operator

Consider the following `if...else`

```
if (alpha < beta)
    min = alpha
else
    min = beta
```

The above can also be achieved by conditional operator

```
min=(alpha<beta) ? alpha : beta;
```

The Conditional Operator

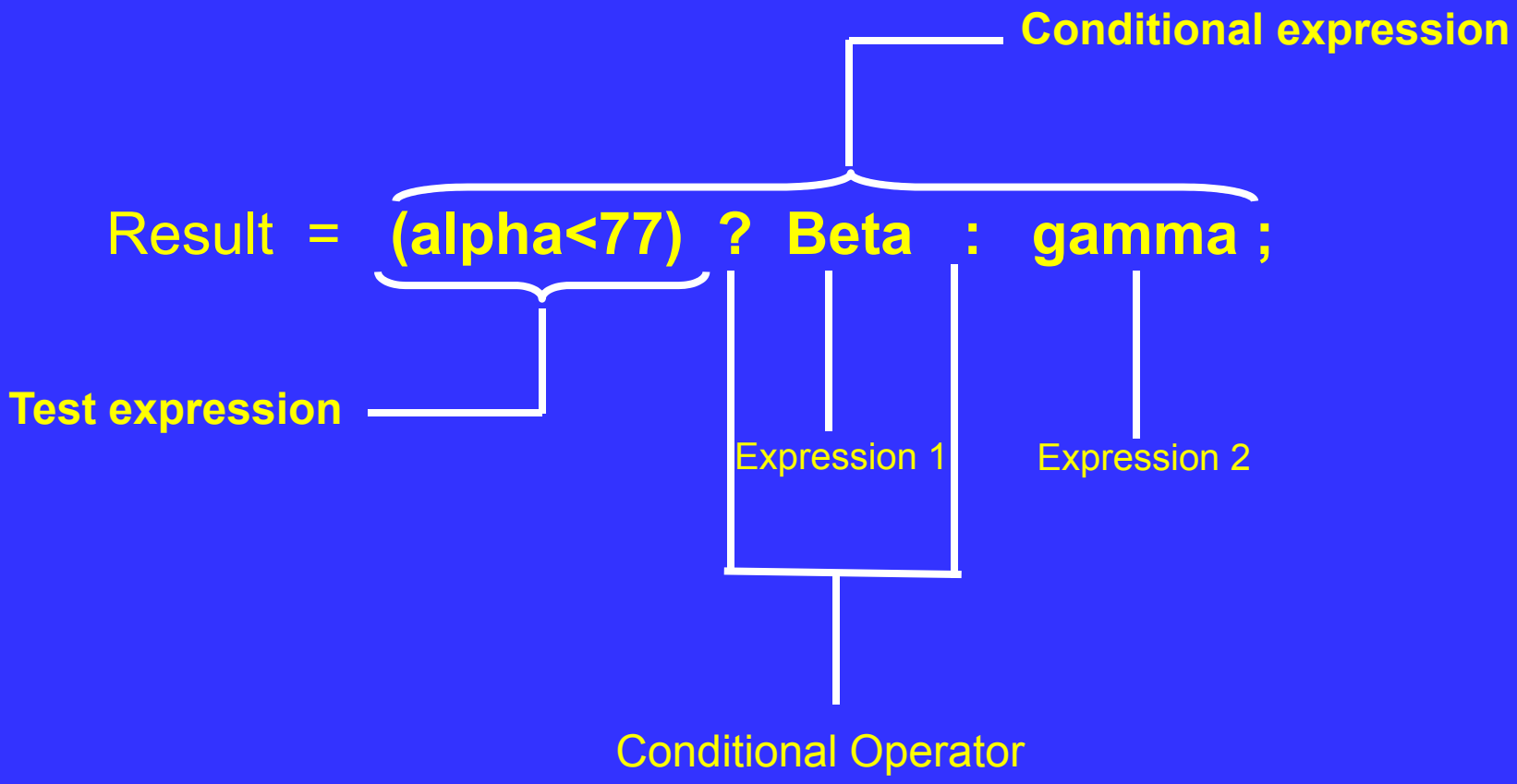
- A variable is given one value if something is true and another value if it's false
- Syntax

  expression1 ? expression2 : expression3

- ***Examples***

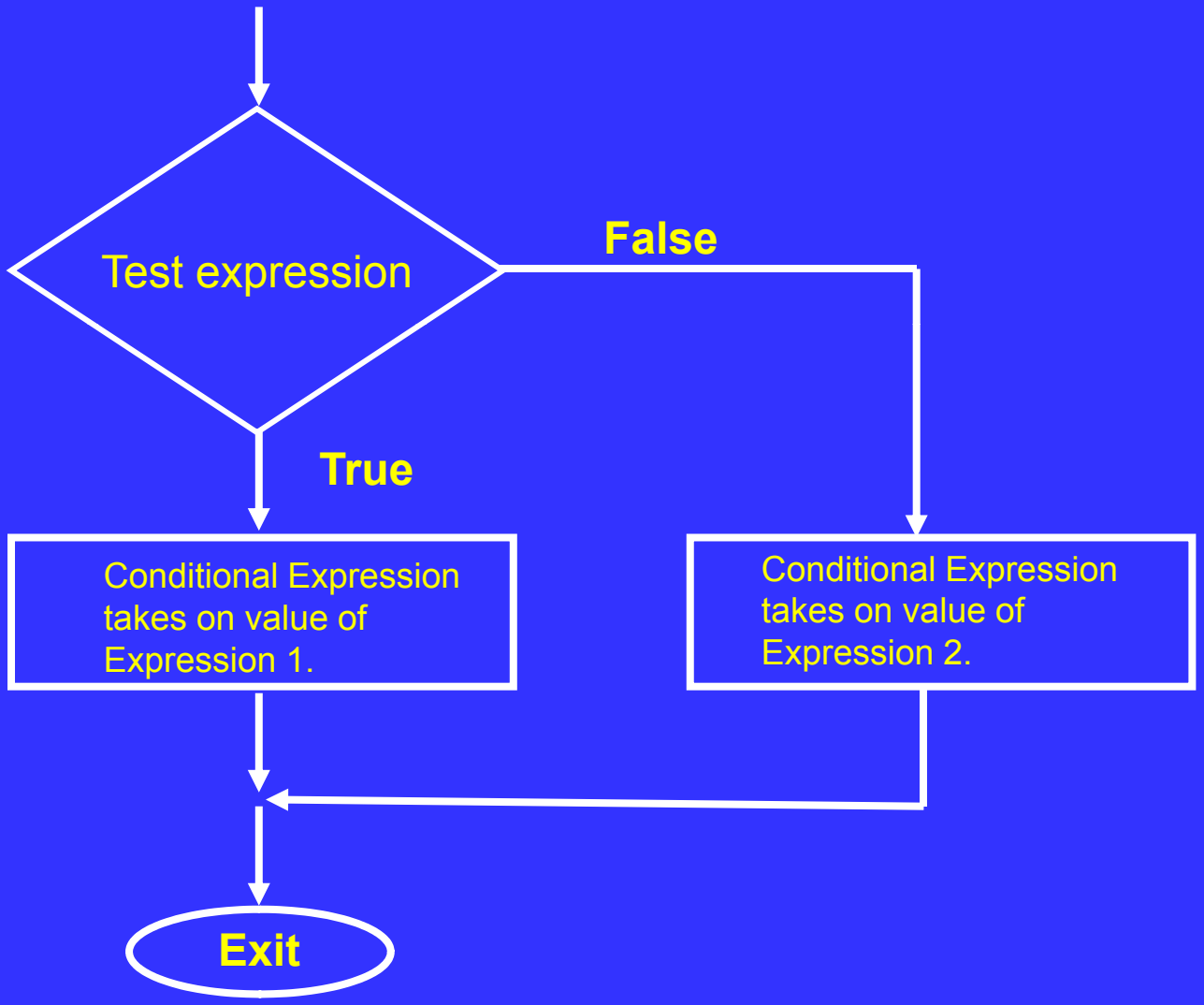
```
min=(alpha<beta) ? alpha : beta;
grade >=60 ? cout<<"Passed" : cout<<"Failed";
cout<<(grade>=60? "passed" : "Failed");
absvalue = n<0 ? -n : n;
```

The Conditional Operator



Syntax of the conditional operator

Operation of the conditional operator



Example : The conditional Operator

```
// condi.cpp
// prints 'x' every 8 columns
// demonstrates conditional operator
#include <iostream.h>


int main()
   {
   for(int j=0; j<80; j++)
      {
      char ch = (j%8) ? ' ' : 'x';
      cout << ch;
      }
   return 0;
   }
```

```
//  for j=0 and every multiple
//  of 8 (j%8) becomes zero (False)
//  so ch is assigned "x", and
//  ' ' (space) otherwise
```

Logical Operators

- Logical AND &&
- Logical OR | |
- Logical NOT !=

- Logical operators combine Boolean variables

Logical and Operator

```
// advenand.cpp
// demonstrates AND logical Operator
#include <iostream.h>

#include <process.h>   //for exit()
#include <conio.h>     //for getche()
int main ()
{
      char dir='a' ;
      int x=10, y=10;
```

Contd.

Logical AND Operator

```
    while( dir != '\r')
    {
        cout << "\nYour location is " << x << " , " << y;
        cout << "\nEnter direction (n, s, e, w): ";
        dir = getche();      //get direction

        switch(dir)
        {
        case 'n': y--; break;   //update coordinates
        case 's': y++; break;
        case 'e': x++; break;
        case 'w': x--; break;
        }   //end switch
        if ( x==7 && y==11)   //if x is 7 and y is 11
              {
              cout << "\nYou found the treasure!\n";
              exit(0);    //exit from program
              }  //end if block
    }   //end while
    return 0;
} //end main
```

Logical AND Operator

- They key to this program is the if statement

  if (x==7 && y==11)
- The test expression will be true only if x is 7 and y is 11.
- Notice that parentheses are not necessary around the relational expressions

  ((x==7) && (y==11)) //inner parentheses not necessary
- Watch out for erroneous indication of endswitch in the book

Logical OR Operator

```
// advenor.cpp
// demonstrates OR logical operator
#include <iostream.h>

#include<process.h>    //for exit()
#include <conio.h>     //for getche()
int main()
{
char dir='a';
int x=10, y=10;
```

Contd.

Logical OR Operator

```
while (dir != '\r')   //quit on Enter key
      {
      cout << "\n\nYour location is " << x << ", " << y ;

      if( x<5 || x>15)   //if x west of 5 OR east of 15
      cout << "\nBeware: dragons lurk here" ;

    cout << "\nEnter direction (n, s, e, w): " ;
    dir = getche() ;
    switch (dir)
          {
          case 'n' : y--; break; //update coordinates
          case 's' : y++; break;
          case 'e' : x++; break;
          case 'w' : x--; break;
          } //end switch
    }   //end while
    return 0;
}   //end main()
```

Logical NOT Operator

- The logical NOT operator is a unary operator – that it takes only one operand
- Examples

`(x==7)`	true if x is equal to 7
`!(x==7)`	true if x is not equal to 7
`x !=7`	true if x is not equal to 7

- Another example

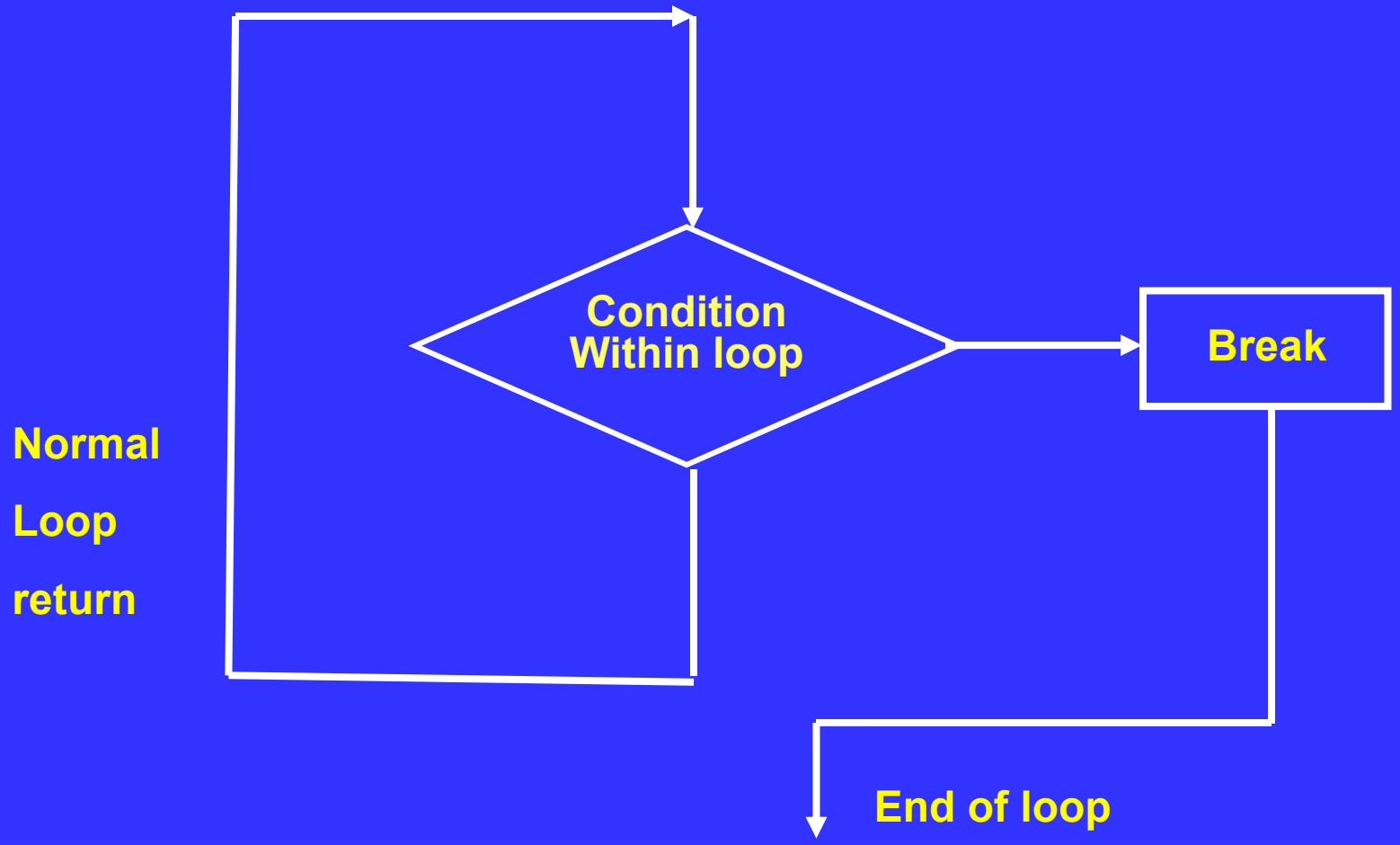
```
if (x%7==0 && y%7==0)
```

is same as

```
if (!(x%7) && !(y%7))
```

The Break Statement

The break statement forces a program out of loop



The Break Statement

```
// showprim.cpp
// displays prime number distribution
#include <iostream.h>
#include <conio.h>      //for getche()
int main()
{
const unsigned char WHITE = 219;        //solid color
   (primes)
const unsigned char GRAY = 176; //gray (non primes)
unsigned char ch;        //for each screen position
```

Contd.

The Break Statement

```
for(int count=0; count<80*25-1; count++)
    {
    ch = WHITE;                    //assume it's prime
    for (int j=2; j<count; j++)            //divide by every integer from
          if(count%j == 0)                 //2 on up; if remainder is 0,
          {
                ch = GRAY;     //it's not prime
                break;         //break out of inner loop
          } // end of if body
    cout << ch;          //display the character
    getch();          //freeze screen until keypress
    } //end of outer for loop
getche();
return 0;
}
```

The continue Statement

It skips the remaining part of the cycle of a loop and takes the control to the start of the loop

Observe the effect of commenting out the continue statement

```
#include <iostream.h>


int main()
{
    int x;
    for (x=0 ; x<=100; x++)
        continue;
        cout << x<<' ';

return 0;
}
```

The continue Statement

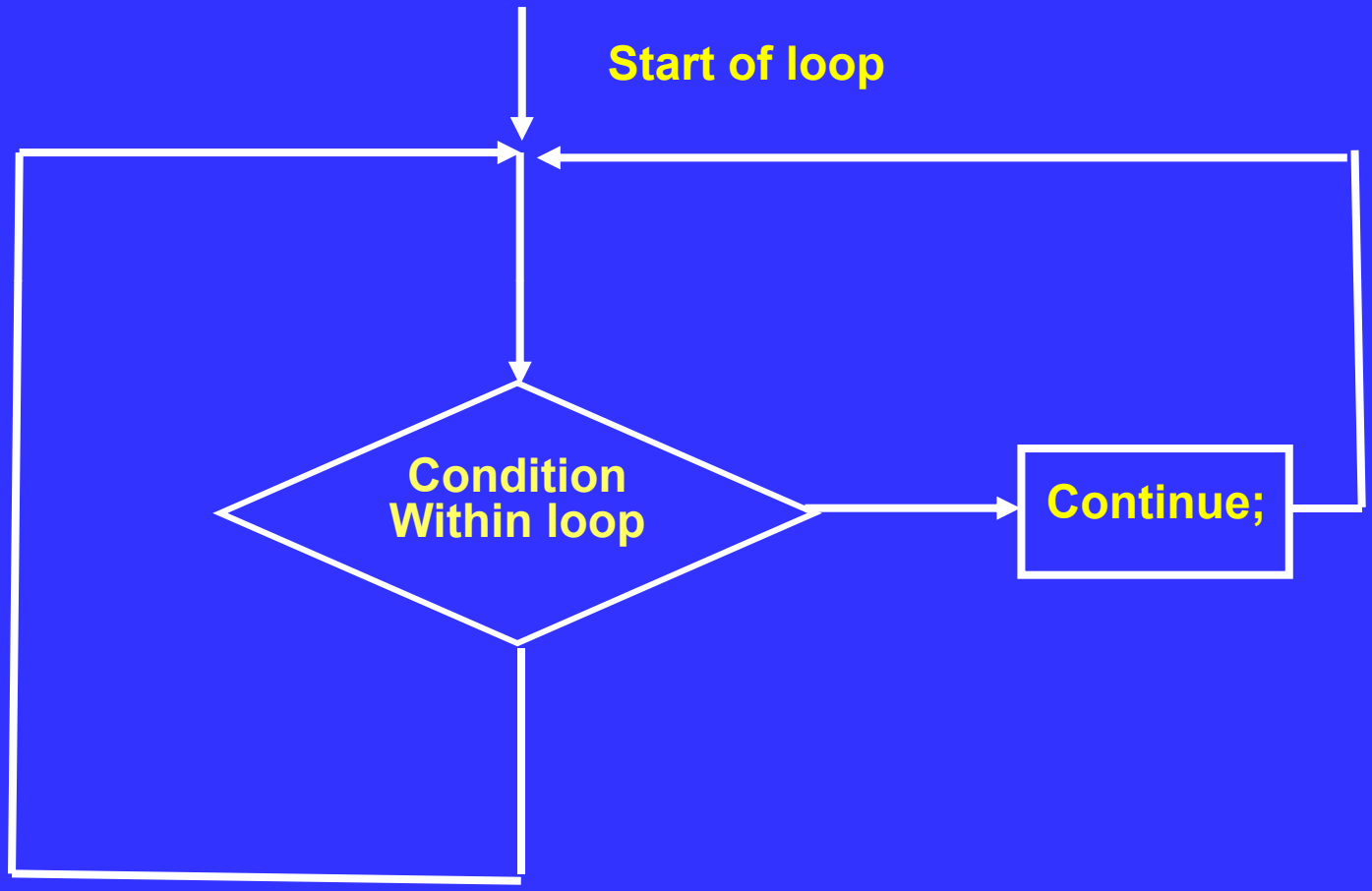
```
// divdo2.cpp
// demonstrates CONTINUE statement
#include <iostream.h>

int main()

{
   long dividend, divisor;
   char ch;
```

Contd.

The continue Statement

```
do
   {
         cout << "Enter dividend: "; cin >> dividend;
         cout << "Enter divisor: ";        cin >> divisor;
         if(divisor == 0)                  //if attempt to
         {                              //divide by 0,
         cout << "Illegal divisor\n";    //display message
         continue;                      // go to top of loop
         }
         cout << "Quotient is " << dividend / divisor;
         cout << ", remainder is " << dividend % divisor;
         cout << "\nDo another? (y/n): ";
         cin >> ch;
   }

   while ( ch != 'n');
   return 0;
}
```

The Continue Statement



goto Statement

- Performs an unconditional transfer of control to the named label. The label must be in the current function.

Example

```
goto SystemCrash;
// other statements
SystemCrash:        // control will begin here
                    // following goto
```

THE END