Conditional Statements

For computer to make decisions, must be able to test CONDITIONS

IF *it is raining*
THEN *I will not go outside*

IF *Count is not zero*
THEN *the Average is Sum divided by Count*

Conditions specified using logical data

Loops

- For
- While
- Do While

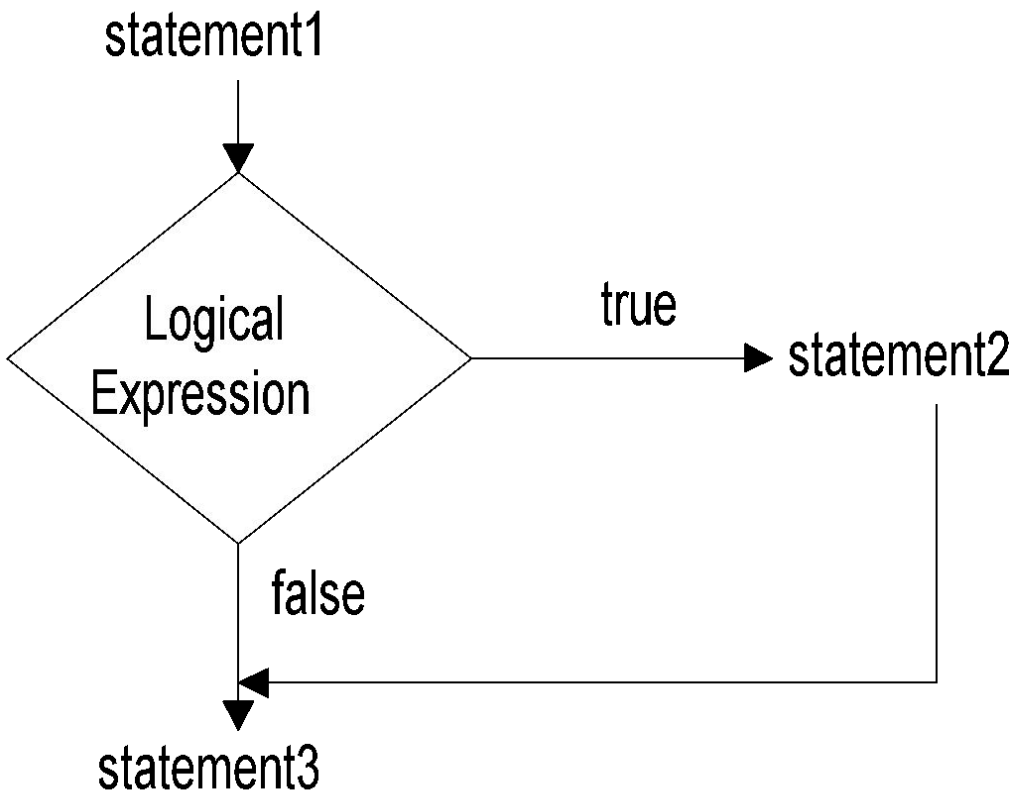
IF Statement

- Syntax: `if` (*LogicalExpr*) *Statement*
- Program context:

  statement1;
  if (LogicalExpr) statement2;
  statement3;
- Order of execution:

LogicalExpr	
1 (true)	0 (false)
statement1	statement1
statement2	statement3
statement3	

Flow of Execution

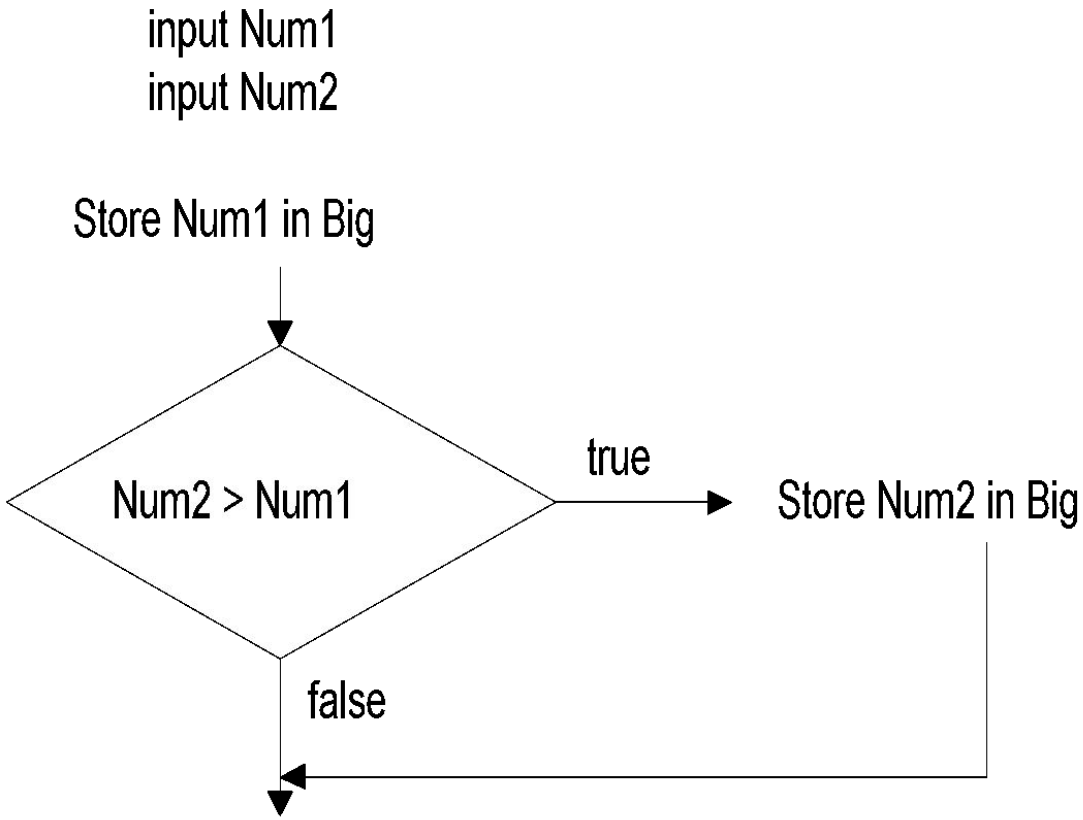


```c
#include<stdio.h>
 int main()
 { int a = 500, b = 100, c;
 if(!a >= 400) b = 300; c = 200;
printf("b = %d c = %d\n", b, c);
return 0; }
```

```c
#include<stdio.h>
int main()
{ int x = 3;
 float y = 3.0;
 if(x == y)
 printf("x and y are equal");
 else printf("x and y are not equal"); return 0;
}
```

```c
#include<stdio.h>
int main()
 { char ch;
 if(ch = printf(""))
printf("It matters\n");
 else printf("It doesn't matters\n");
 return 0;
}
```

Flow Chart for our Problem



Code for Solution

```
int findMax(int Num1, int Num2) {
   int Big;

   Big = Num1;
   if (Num2 > Num1) Big = Num2;
   return Big;
}
```

Write a program to print even nos from 1 to 100

- #include <stdio.h>

```
int main() {
 for (int i = 1; i <= 100; i++) {
  if (i % 2 == 0) {
      printf("%d ", i);     }
 }
 return 0;
}
```

Write a program to print factorial of a number

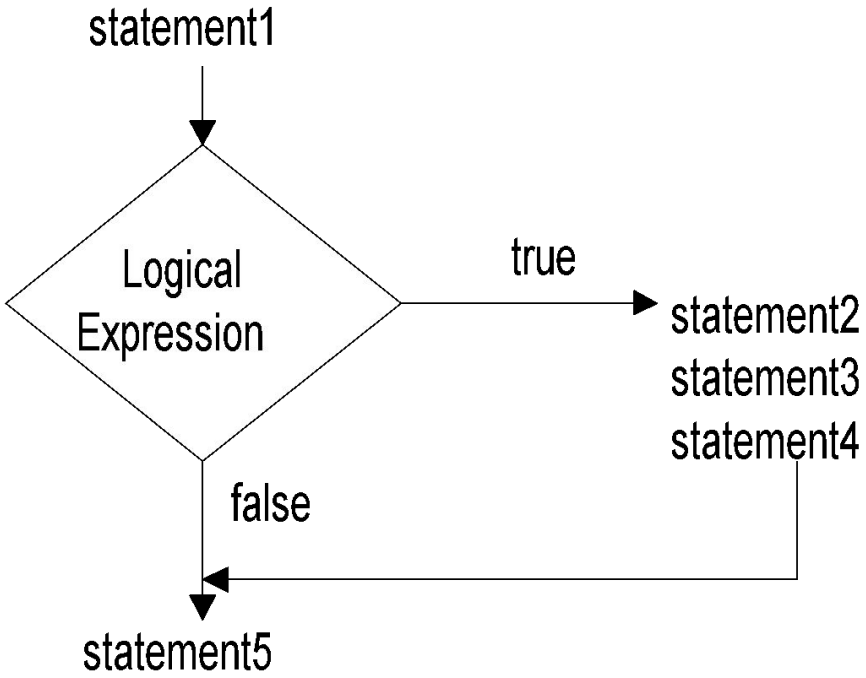
```
#include <stdio.h>
int main() {
 int n;
 unsigned long factorial = 1;
scanf("%d", &n);
for (int i = 1; i <= n; i++) {
     factorial *= i;
     }
printf("Factorial of %d = %lu\n", n, factorial);
 }

   return 0;
}
```

Check if num is prime or not

```
int main() {
 int num;
bool isPrime = true;
scanf("%d", &num);
for (int i = 2; i <= num - 1; i++) {
            if (num % i == 0) {
                isPrime = false;
                  break; }
       }
   if (isPrime) {
       printf("%d is a prime number.\n", num);
  } else {
      printf("%d is not a prime number.\n", num);
   }
 return 0;
```

Executing > 1 Statement in an If

- What if you want to execute > 1 stmt in If?



> 1 Stmt in an If

Does this work?

```
Statement1;
If (LogicalExpr)
  Statement2;
  Statement3;
  Statement4;
Statement5;
```

P

No, the indenting is irrelevant, section P is:

```
If (LogicalExpr)
  Statement2;
Statement3;
Statement4;
```

> 1 Stmt in an If (A solution)

Solution - use a compound statement:

```
Statement1;
If (LogicalExpr) {
  Statement2;
  Statement3;
  Statement4;
}
Statement5;
```

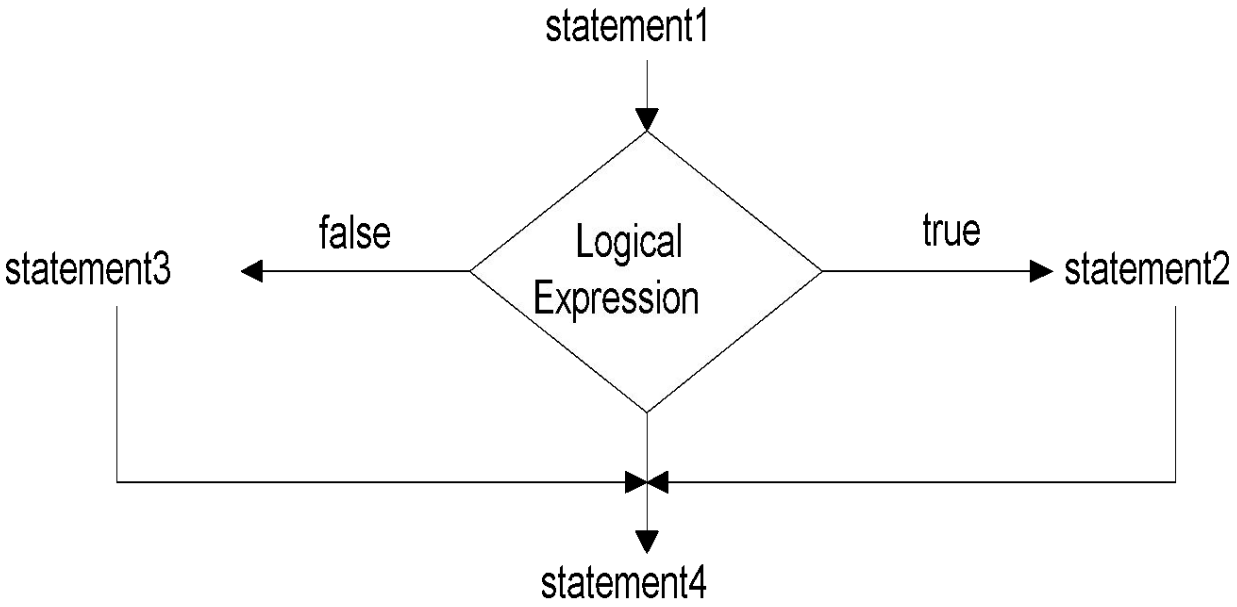
Statement in If

Any statement reasonable as the single statement in if

- expression statement
  - assignment statement
  - function call (printf, scanf, etc.)
- compound statement
- if statement

Two-Way Selection

- Sometimes desirable for statement to be executed when Logical Expression false

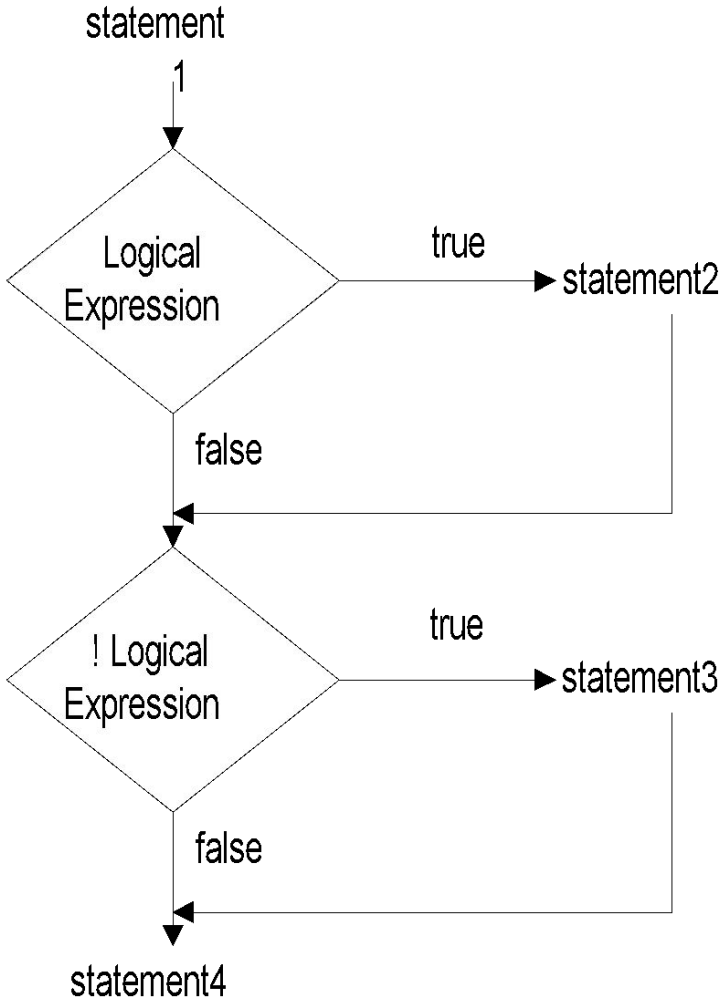


Two-Way Selection with If

- Possible to use combinations of if:

```
Statement1;
if (LogicalExpr)
  Statement2;
if (!(LogicalExpr))
  Statement3;
Statement4;
```

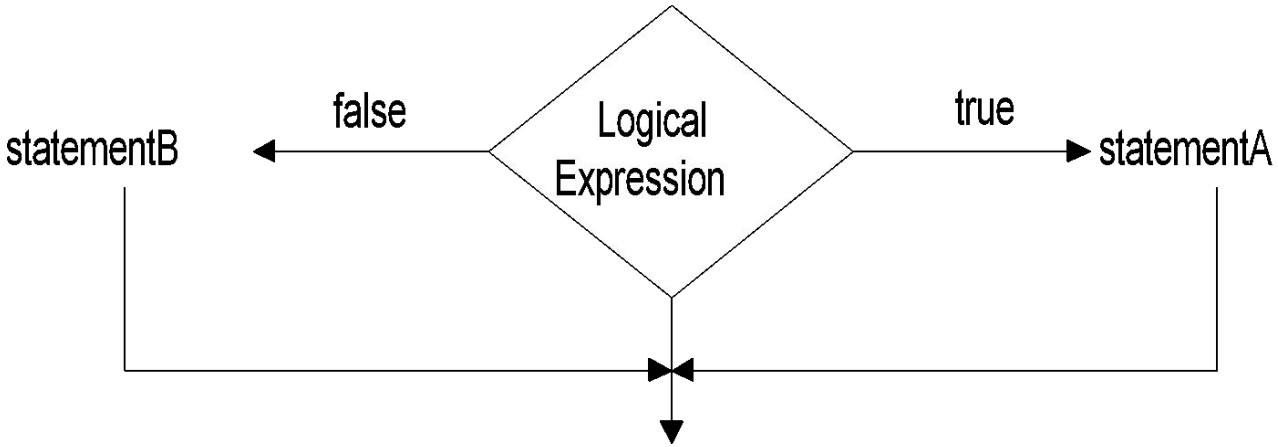
- But not efficient



If-Else Statement

- Syntax:

```
if (LogicalExpr)
   StatementA
else
   StatementB
```



FindMax with If-Else

```
int findMax(int Num1, int Num2) {
  int Big;

  Big = Num1;                       /* 1 assign + */
  if (Num2 > Num1) Big = Num2; /* 1 test + */
  return Big;                       /* maybe 1 assign */
} /* test plus 1 or 2 assigns */

int findMax(int Num1, int Num2) {
  int Big;

  if (Num2 > Num1)  /* 1 test + */
    Big = Num2;     /* 1 assign or */
  else
    Big = Num1;     /* 1 assign */
  return Big;
} /* test plus 1 assign */
```

Using If-Else for Robustness

```
float calcAverage(float sum,
   int count) {
   if (count == 0)
      return 0.0;
   else
      return sum / count;
}
```

- Note return statement for each condition

Compound Statements and If-Else

```
if ((year % 4) == 0) {
  printf("Leap year\n");
  numDays = 366;
}
else {
  printf("Not a leap year\n");
  numDays = 365;
}
```

Programming Tip: Compound Stmt

- Does not hurt to always use compound statements for if, if-else statements

```
if (LogicalExpr) {
  /* statement or statements */
}

if (LogicalExpr) {
  /* statement(s) */
}
else {
  /* statement(s) */
}
```

- Easy to add statements later

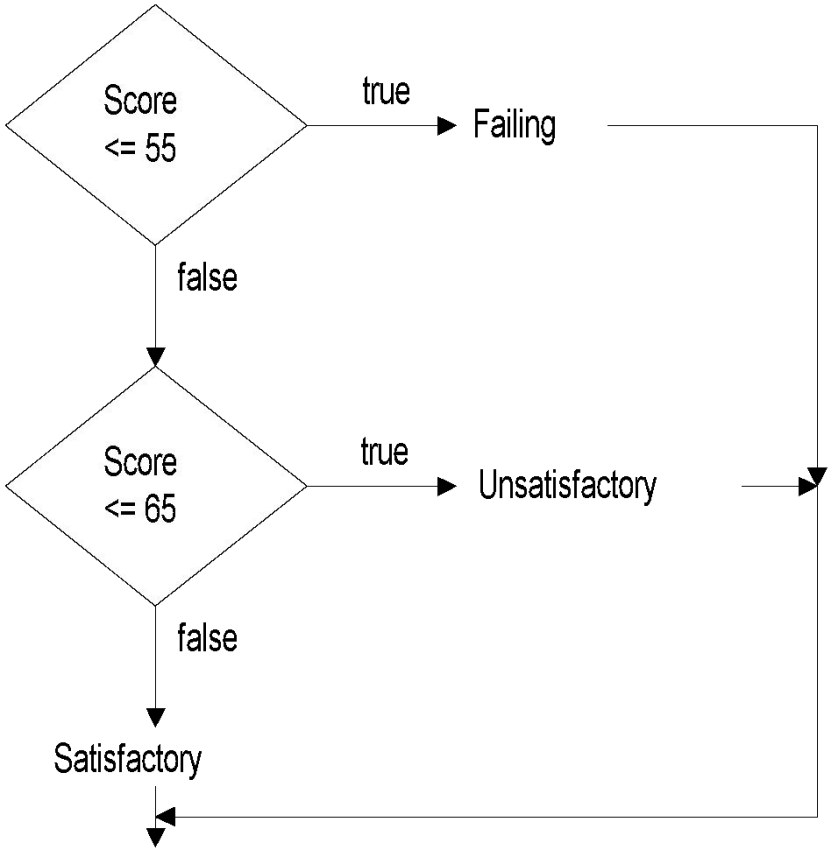
Multiway Selection

- Multiway if more than 2 alternatives
- Example:

Student Score	Message
0-55	Failing
56-65	Unsatisfactory
66-100	Satisfactory

- If-Else has two alternatives, to do multiway, string together If-Else statements

Multiway Flow Chart



Multiway with If-Else

```
if (score <= 55)
  printf("Failing\n");
else
  if (score <= 65)
    printf("Unsatisfactory\n");
  else
    printf("Satisfactory\n");
```

Switch Statement

- More readable approach: switch statement

```
switch (score) {
  case 9: case 10:
    grade = 'A';
    break;
  case 7: case 8:
    grade = 'B';
    break;
  case 5: case 6:
    grade = 'C';
    break;
  case 0: case 1: case 2: case3: case 4:
    grade = 'F';
}
```

Switch Format

```
switch (Expression) {
   case const1-1: case const1-2: ...
      statement
      statement
      ...
   case const2-1: case const2-2: ...
      statement
      statement
      ...
   default: ...
      statement
      statement
      ...
}
```

Switch Rules

- Expression must be integral type (no float)
- Case labels must be constant values
- No two case labels with same value
- Two case labels can be associated with same set of statements
- Default label is not required
- At most one default label

Executing Switch Example

```
switch (score) {
  case 9: case 10:
    grade = 'A';
  case 7: case 8:
    grade = 'B';
  case 5: case 6:
    grade = 'C';
}
```

- score is 9:

  grade = 'A'

  grade = 'B'

  grade = 'C'

- score is 7:

  grade = 'B'

  grade = 'C'

- score is 5:

  grade = 'C'

- not quite what we want

The break Statement

- break used to indicate a set of statements is finished (and no more should be executed)
- syntax: `break;`
- break says to stop executing and go to the next } (skipping any statements in between)
- add after each set of cases

default case

The default case can be used to deal with robustness issues (score is < 0 or > 10)

```
switch (score) {
  case 9: case 10:
    grade = 'A';
    break;
  case 7: case 8:
    grade = 'B';
    break;
  case 5: case 6:
    grade = 'C';
    break;
  case 0: case 1: case 2: case3: case 4:
    grade = 'F';
    break;
  default:
    printf("Bad score %d\n",score);
}
```

Other Expression Types

- Any integral type can be used as expression
- Cases must match

```
char married;
switch (married) {
  case 'S': case 's':
    printf("single"); break;
  case 'D': case 'd':
    printf("divorced"); break;
  case 'M': case 'm':
    printf("married");
}
```

When Not to Use Switch

- Example: Case 0 <= X <= 100

```
switch (x) {
   case 0: case 1: case 2: case 3:
   case 4: case 5: case 6: case 7:
   …
   case 100:
…
}
```

- Better to use nested ifs

Ternary (or Conditional) Operator

- Represented as ?:
- Int a=(b>c)? b:c

```
#include<stdio.h> int main()
 { char str='C';
 int a = 5;
 printf(a >10?"Ps\n":"%c\n", str);
 return 0; }
```