Control Statements

Indranil Sen Gupta

Dept. of Computer Science & Engg.
Indian Institute of Technology
Kharagpur

What do they do?

- Allow different sets of instructions to be executed depending on the outcome of a logical test.
  - Whether TRUE or FALSE.
  - This is called *branching*.
- Some applications may also require that a set of instructions be executed repeatedly, possibly again based on some condition.
  - This is called *looping*.

How do we specify the conditions?

- Using relational operators.
  - Four relation operators: <, <=, >, >=
  - Two equality operators: ==, !=
- Using logical operators / connectives.
  - Two logical connectives: &&, ||
  - Unary negation operator: !

Examples

```
count <= 100
(math+phys+chem)/3 >= 60
(sex='M') && (age>=21)
(marks>=80) && (marks<90)
(balance>5000) || (no_of_trans>25)
!(grade='A')
!((x>20) && (y<16))
```

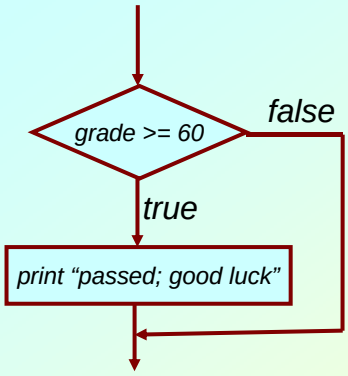
The conditions evaluate to …

- Zero
  - Indicates *FALSE*.
- Non-zero
  - Indicates *TRUE*.
  - Typically the condition TRUE is represented by the value ‘1’.

Branching: The if Statement

- Diamond symbol (decision symbol) - indicates decision is to be made.
  - Contains an expression that can be TRUE or FALSE.
  - Test the condition, and follow appropriate path.
- Single-entry / single-exit structure.
- General syntax:

  if (condition) { …….. }
  - If there is a single statement in the block, the braces can be omitted.



A decision can be made on any expression.

zero - false

nonzero - true

```
if (grade>=60)
{
  printf(“Passed \n”);
  printf(“Good luck\n”);
}
```


Example

```
#include <stdio.h>
main()
{
    int  a,b,c;
    scanf (“%d %d %d”, &a, &b, &c);
    if ((a>=b) && (a>=c))
        printf (“\n The largest number is: %d”, a);
    if ((b>=a) && (b>=c))
        printf (“\n The largest number is: %d”, b);
   if ((c>=a) && (c>=b))
        printf (“\n The largest number is: %d”, c);
}
```

Confusing Equality (==) and Assignment (=) Operators

- **Dangerous error**
  - Does not ordinarily cause syntax errors.
  - Any expression that produces a value can be used in control structures.
  - Nonzero values are true, zero values are false.
- **Example:**

```
if (payCode = = 4)
   printf( "You get a bonus!\n" );

 if (payCode = 4)
   printf( "You get a bonus!\n" );
```

WRONG

Some Examples

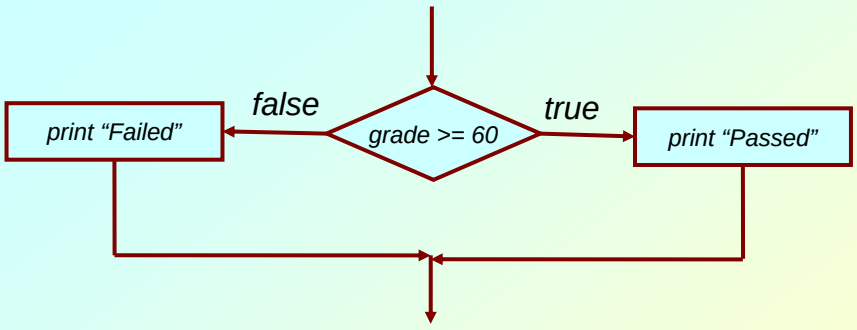
```
if (10<20)  { a = b + c;  printf (“%d”, a);  }

if ((a>b) && (x=10))  { ……. }

if (1)  { …… }

if (0)  { …… }
```

Branching: The if-else Statement

- Also a single-entry / single-exit structure.
- Allows us to specify two alternate blocks of statements, one of which is executed depending on the outcome of the condition.
- General syntax:

```
if  (condition)   { …… block 1 ……. }
else  { ……. block 2 ……. }
```

  - If a block contains a single statement, the braces can be deleted.




```
if ( grade >= 60 )
      printf ("Passed\n");
else
      printf ("Failed\n");
```

Nesting of if-else Structures

- It is possible to nest if-else statements, one within another.
- All if statements may not be having the "else" part.
  - Confusion??
- Rule to be remembered:
  - An "else" clause is associated with the closest preceding unmatched "if".
  - Some examples shown next.


```
if e1 s1
else if e2 s2

if e1 s1
else if e2 s2
else s3

if e1 if e2 s1
else s2
else s3

if e1 if e2 s1
else s2
```

?

```
if e1 s1                          if e1 s1
else if e2 s2                     else if e2 s2

if e1 s1                          if e1 s1
else if e2 s2                     else if e2 s2
else s3                                else s3

if e1 if e2 s1                    if e1 if e2 s1
else s2                                else s2
else s3                           else s3

if e1 if e2 s1                    if e1 if e2 s1
else s2                                else s2
```


Example

```
#include <stdio.h>
main()
{
    int  a,b,c;
    scanf (“%d %d %d”, &a, &b, &c);
    if (a>=b)
       if (a>=c)
          printf (“\n The largest is: %d”, a);
       else     printf (“\n The largest is: %d”, c);
    else
      if (b>=c)
          printf (“\n The largest is: %d”, b);
      else     printf (“\n The largest  is: %d”, c);
}
```

Example

```
#include <stdio.h>
main()
{
    int  a,b,c;
    scanf (“%d %d %d”, &a, &b, &c);
    if ((a>=b) && (a>=c))
        printf (“\n Largest number is: %d”, a);
    else if (b>c)
            printf (“\n Largest number is: %d”, b);
         else
            printf (“\n Largest number is: %d”, c);
}
```


The Conditional Operator ? :

- This makes use of an expression that is either true or false. An appropriate value is selected, depending on the outcome of the logical expression.
- Example:

```
interest = (balance>5000) ? balance*0.2 : balance*0.1;
```

Returns a value

- Examples:

```
x = ((a>10) && (b<5)) ? a+b : 0

(marks>=60) ? printf("Passed \n") : printf("Failed \n");
```


The switch Statement

- This causes a particular group of statements to be chosen from several available groups.
  - Uses "switch" statement and "case" labels.
  - Syntax of the "switch" statement:

```
switch (expression)  {
  case expression-1: { …… }
  case expression-2: { …… }

  case expression-m: { …… }
  default: { …… }
}
```

where "expression" evaluates to int or char

Example

```
switch (letter)
{
  case 'A':
      printf ("First letter \n");
      break;
  case 'Z':
      printf ("Last letter \n");
      break;
  default :
      printf ("Middle letter \n");
      break;
}
```


The break Statement

- Used to exit from a switch or terminate from a loop.
  - Already illustrated in the previous example.
- With respect to “switch”, the “break” statement causes a transfer of control out of the entire “switch” statement, to the first statement following the “switch” statement block.

Example

```
switch  (choice = getchar())  {
  case 'r':
  case 'R': printf ("RED \n");
            break;
  case 'g':
  case 'G': printf ("GREEN \n");
            break;
  case 'b':
  case 'B': printf ("BLUE \n");
            break;
  default:  printf ("Invalid choice \n");
}
```


Example

```
switch  (choice = toupper(getchar()))  {
  case 'R': printf ("RED \n");
            break;
  case 'G': printf ("GREEN \n");
            break;
  case 'B': printf ("BLUE \n");
            break;
  default:  printf ("Invalid choice \n");
}
```

```
int main () {
  int operand1, operand2;
  int result = 0;
  char operation;
  /* Get the input values */
  printf (“Enter operand1 :”);
    scanf(“%d”, &operand1) ;
  printf (“Enter operation :”);
    scanf (“\n%c”, &operation);
  printf (“Enter operand 2 :”);
    scanf (“%d”, &operand2);
  switch (operation)    {
  case ‘+’ :
    result = operand1 + operand2;
    break;
  case ‘-’ :
    result = operand1 - operand2;
    break;
  case ‘*’ :
    result = operand1 * operand2;
    break;
  case ‘/’ :
    if (operand2 !=0)
     result = operand1 / operand2;
    else
     printf (“Divide by 0 error”);
    break;
  default:
   printf (“Invalid operation\n”);
}
  printf (“Result: %d\n”,result);
}
```


- **The “switch” statement also constitutes a single-entry / single-exit structure.**