## Relational/Equality Operations

- Relationship between 2 variables or expressions.
- Returns a numerical value depending on the result.
- 1 indicates a TRUE and 0 indicates a FALSE.
- Relational operators >, <, >=, <=
- Equality operators ==, !=

**RELATIONAL/EQUALITY OPERATORS**

| SYMBOL | PURPOSE |
|---|---|
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| == | Equal to |
| != | Not equal to |

**Note:**

**=**

- Denotes assignment
- Result of operation in right hand side is assigned to the variable mentioned in the left hand side.
- **x = 5;  /* returns 5 to x*/**

**==**

- Denoted equal to.
- Returns either 1 or 0.
- **x == 6;  /* returns 0 */**

## The if Statement

- For Decision-making capabilities
- Syntax:

```
If(conditions){
    /* body of if statement appears here!*/
}
```

Description: If the condition or conditions are true then the body of the "if" statement is executed.

*Two Forms of "if" Statement*

1- Single Statement

- Body of "if" only includes ONE statement
- No need for braces to mark the body of the "if". But if used is correct.

- Syntax:

```
if(condition)
        statement1;
statement2;
```

Note: statement2 is NOT part of the "if" statement and will always be executed regardless of the status of the condition.

```
/* USE OF IF STATEMENT */
#include <stdio.h>
main( ){
        float   MyGPA;
        float   MinGPA=2.0;

        /* Input the GPA */
        printf("Enter your GPA: ");
        scanf("%f",&MyGPA);
        /* Test the value of GPA and print the result */
        if (MyGPA > MinGPA)
                printf("Your status might be clear.\n");
        printf("Your GPA is %f.\n", MyGPA);
}
```

**1st printf**

**2nd printf**

Note: The body of the if is not marked by braces, hence, it contains only one statement that is the 1st printf statement and the 2nd printf statement is outside the if body and is always executed.

2- Compound Statement

- Body of "if" includes more than one statement.
- Must include braces, { }, to enclose the body of the "if".
- Syntax:

```
if (conditions)
{
        statement_1;
        statement_2;
        …
}
statement_N;
```

Description: Conditional statement.

- If the condition is TRUE then all the statements within the braces are executed.
- Otherwise those statements are not being executed.
- statement_N is NOT part of the "if" statement and will always be executed regardless of the condition status.

Example:

```
/* COMPOUND “if” STATEMENT */
/* Are you getting A+ in your course? */
#include <stdio.h>
main( )
{
        float           Grade, GPA;

        /* Input  the grade */
        printf("Please enter your grade: ");
        scanf("%f", &Grade);

        /* Test if the grade is A+ and print the result */
        if (Grade >= 90.0)
        {
                printf("You're getting A+ in this course!\n");

                /* Input the GPA  */
                printf("Please enter your GPA: ");
                scanf("%f",&GPA);

                printf("Your GPA is %f\n",GPA);
        }
        printf("Have a good day!\n\n\n");
}
```

**If…else…**

- Similar to the “if” statement seen before.
- Adds to the decision making capability.
- General Syntax:

```
if   (condition)
{
    statement1;
 }
else {
    statement2;
 }
```

- If the condition is TRUE then statement1 is executed.
- Otherwise statement2 is executed.

Example:

```
/* PROGRAM TO CALCULATE COURSE MARKS */
#include <stdio.h>
```

```
#define     MidTermWeight       0.30
#define     ExamWeight          0.50
#define     LabWeight           0.20

main( ){
        float       MidTerm, Exam, Lab;
        float       WrittenWork, LabWork, Grade;

        /* Input the mid-term, exam and lab mark */
        printf("Please enter your mid-term mark: ");
        scanf("%f",&MidTerm);

        printf("Please enter your exam mark: ");
        scanf("%f",&Exam);

        printf("Please enter your lab mark: ");
        scanf("%f",&Lab);

        /* Calculate the grade, test if the grade is A+ and print the result */
        WrittenWork = MidTerm * MidTermWeight + Exam * ExamWeight;
        LabWork     = Lab * LabWeight;
        Grade       = WrittenWork + LabWork;

        if (Grade >= 90)
                printf("You got definitely A+.\n");
        else
                printf("Either B, C, D, or even F!.\n");
}
```

**#define**

- Can be used to indicate a constant.
- Reduces repetition of numbers.
- If the definition is changed then all values in the program are changed.
- Semicolon is not required.
- i.e.

```
#include        HST         100
```

Here HST represents the 100

**Nested if…else**

- Very similar to if … else … statement seen before
- Syntax:

```
if      (condition_1)
        DoThis_1;
else
        if (condition _2)
                DoThis _2;
        else
                DoThis _3;
```

Description:
If condition _1 is TRUE then DoThis _1
If condition _1 is FLASE and condition _2 is TRUE then DoThis _2
If condition _1 is FLASE and condition _2 is FALSE then DoThis _3

Example:

```
/*********************************************************
** PROGRAM TO CALCULATE GST AND PST
** SHOWS HOW NESTED "if…else" FUNCTION.
*********************************************************/
#include <stdio.h>
#include <stdlib.h>        /* Adding <stdlib.h> library to include exit( )*/
#define     GST         0.07
#define     PST         0.08
main( ){
        char        choice;
        float       Ptax, Gtax;
        float       BTprice, ATprice;
        /* Get the selected letter by user */
        printf("Use the menu - Select a Letter indicating which tax calculation
you want:\n");
        printf("P represents Provincial Tax and G represents Goods &
Services Tax.\n");
        puts("Now enter your choice (P or G):");
        scanf("%c",&choice);

        /* Get the price before tax, calculate GST and PST print the result*/
        printf("Please enter the Before Tax price\n");
        scanf("%f",&BTprice);

        if (choice == 'G'){
                ATprice =  GST * BTprice + BTprice;
```

```
            printf("The Before Tax Price is %f.\n",BTprice);
            printf("The After Tax Price is %f.\n", ATprice);
        }
        else
            if (choice == 'P'){
                ATprice =  PST * BTprice + BTprice;
                printf("The Before Tax Price is %f.\n",BTprice);
                printf("The After Tax Price is %f.\n", ATprice);
            }
          else{
                printf("That was not one of the choices.\n");
                printf("Sorry!\n");
            }
}
```

**Logical Operations – AND, OR, NOT**

- Logical AND is represented by "&&"
- i.e.
  (expression1 ) && (expression2 )

Description: This operation is TRUE only if expression1 is TRUE and expression2 is TRUE.
Works based on the AND truth table as follows:

**LOGICAL AND TRUTH TABLE**

| **EXPRESSION 1** | **EXPRESSION 2** | **AND** |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Note: "1" means TRUE and "0" means FALSE.
Note: Any nonzero value is treated as TRUE

**Logical OR Operation**

- Logical OR is represented by "||"
- i.e. (expression1) || (expression2)

Description: This operation is FALSE only if both expression1 and expression2 are FALSE.
Works based on the OR truth table as follows:

**LOGICAL OR TRUTH TABLE**

| EXPRESSION 1 | EXPRESSION 2 | OR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Logical NOT Operation

- Logical NOT is represented by "!"
- i.e. ! (expression1)

Description: This operation negates the logical value of the expression. Works based on the NOT truth table as follows:

**LOGICAL NOT TRUTH TABLE**

| EXPRESSION 1 | NOT |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Precedence C Operators/Mathematical/Relational, Logical

**PRECEDENCE FOR C OPERATORS**

| HIGH ↓ LOW | OPERATION | PURPOSE |
|---|---|---|
| | ! | Logical Not |
| | *, / | Multiplication and Division |
| | +, - | Addition and Subtraction |
| | <, <=, >=, > | Less, Less or Equal, Greater or Equal, Greater |
| | ==, != | Equal, Not Equal |
| | && | Logical AND |
| | \|\| | Logical OR |