

# COSC1101 – Programming Fundamentals

Maham Khan

Lecture - 7

Special Thanks:

To

Dr. Sadaf Tanveer, Assistant Professor

CIIT Islamabad Campus

for slide Material

# Relational Operators

< > <= >= == !=

= is the assignment operator

```
void main()
```

```
{  
    int i=1;  
    clrscr();  
    printf("Is i less than 0? %d\n",i==0);  
    printf("Is i greater than 0? %d\n",i==1);  
    getch();  
}
```

Output

Is i less than 0? 0

Is i greater than 0? 1

# Relational Operators

```
void main()
{
    int i=1;
    clrscr();
    printf("Is i less than 0? %d\n",i<0);
    printf("Is i greater than 0? %d\n",i>0);
    getch();
}
```

## Output

Is i less than 0? 0

Is i greater than 0? 1

# Precedence

```
void main()
```

```
{
```

```
    clrscr();
```

```
    printf("Ans: %d\n",2+1<4);
```

```
    getch();
```

```
}
```

Output 1

```
void main()
```

```
{
```

```
    printf("Ans: %d\n",1<2+4);
```

```
    getch();
```

```
}
```

Output 1

# Decision Constructs

The if statement

```
void main()  
{  
    char ch;  
    clrscr();  
    ch=getch();  
    if (ch=='y')  
        printf("You typed y.");  
    getch();  
}
```

Output

y

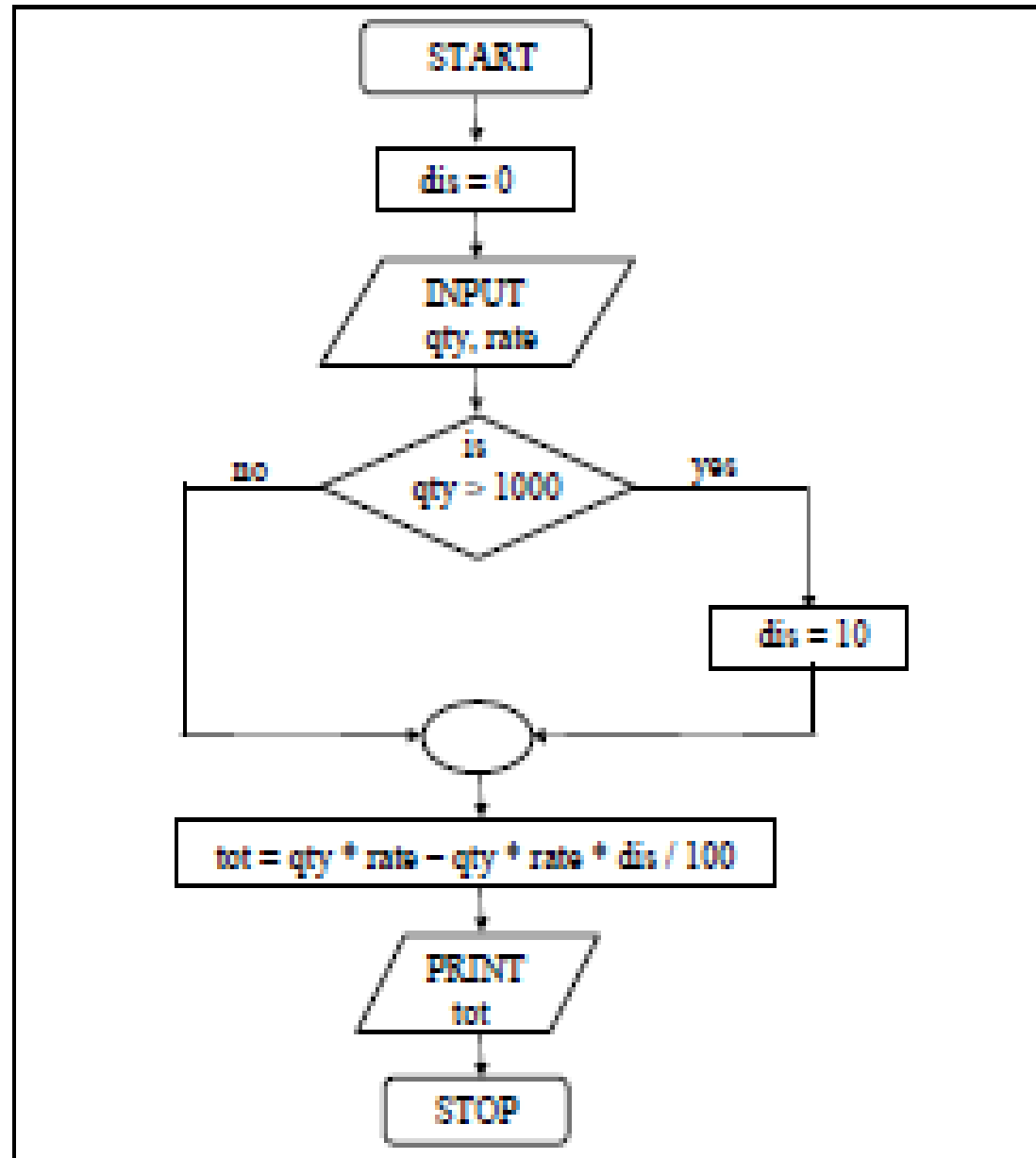
You typed y

# Practice Exercise

While purchasing certain items, a store offers a discount of 10% if the quantity purchased is more than 1000.

Write a program that input quantity and price per item through the keyboard, and calculate the amount to be charged from the customers.

# Flowchart





# Class Exercise

```
/* Calculation of payment to be charged from customers */  
void main( )  
{  
    int qty, discount = 0 ;  
    float rate, total ;  
    printf ( "Enter quantity and rate " ) ;  
    scanf ( "%d %f", &qty, &rate) ;  
    if ( qty > 1000 )  
        discount = 10 ;  
    total = ( qty * rate ) - ( qty * rate * discount / 100 ) ;  
    printf ( "Total expenses = Rs. %f", total) ;  
}
```

# Class Exercise

## Output

Enter quantity and rate 1200 10.0

Total expenses = Rs. 10,800.0000

Enter quantity and rate 200 15.0

Total expenses = Rs. 3000.0000

# Multiple Statements within if

```
if ( year_of_service > 3 )  
{  
    bonus = 2500 ;  
    printf ( "Bonus = Rs. %d", bonus ) ;  
}
```

# The if-else statement

When **if** condition evaluates to true, the group of statements under **if** are executed

It does nothing when the expression evaluates to false

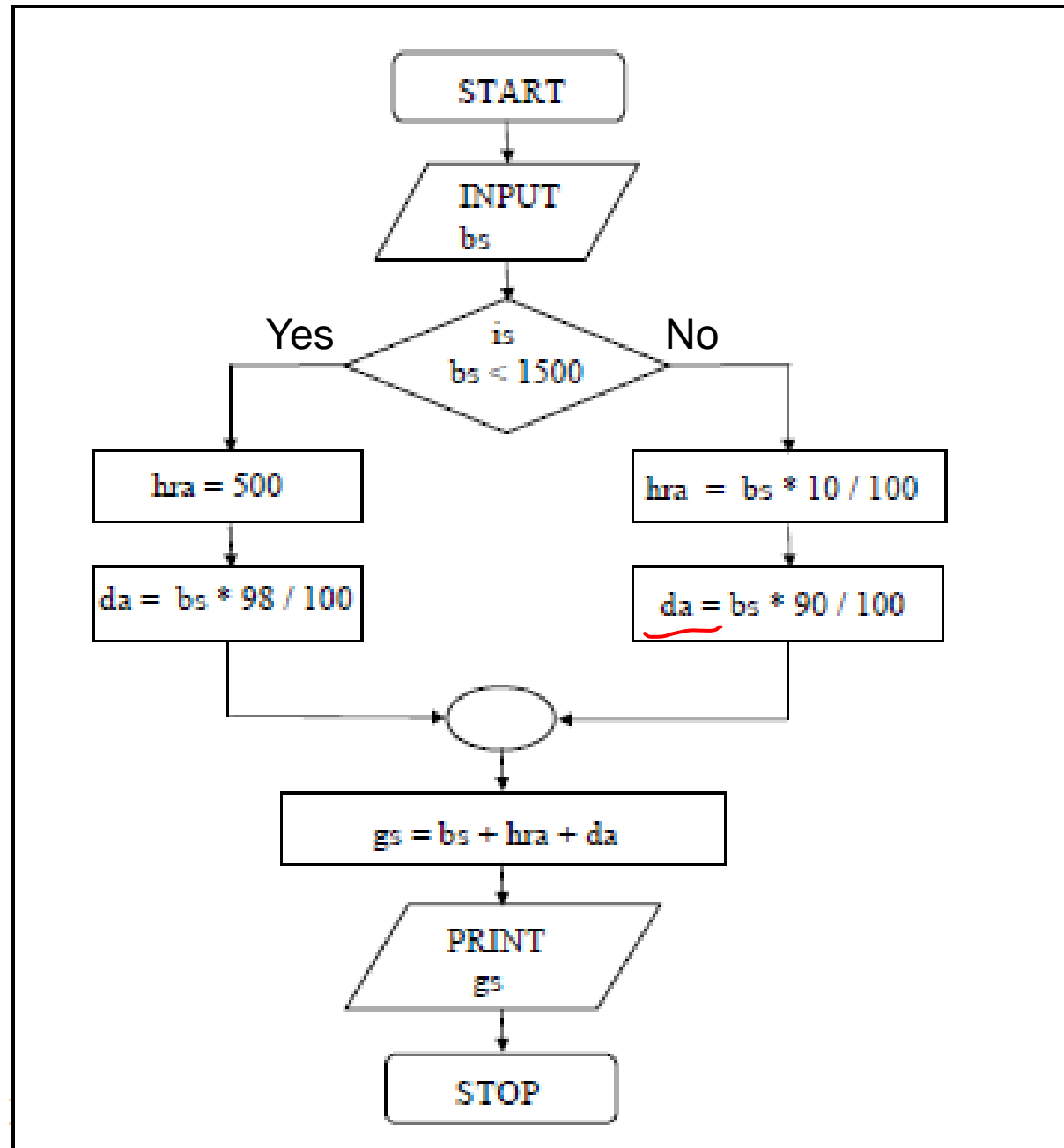
We can do something when the **if** condition evaluates to false using the **else** statement

# Practice Exercise

In a company an employee pay is computed as under:

- If his basic salary is less than Rs. 1500, then  $HRA = 10\%$  of basic salary and  $DA = 90\%$  of basic salary. If his salary is greater than or equal to Rs. 1500, then  $HRA = Rs. 500$  and  $DA = 98\%$  of basic salary.
- If the employee's salary is input through the keyboard write a program to find his gross salary.

# Flowchart



```
main( )
{
    float basic, grs, da, hra ;
    printf ( "Enter basic salary " ) ;
    scanf ( "%f", &basic ) ;
    if ( basic < 1500 )
    {
        hra = basic * 10 / 100 ;
        da = basic * 90 / 100 ;
    }
    else
    {
        hra = 500 ;
        da = basic * 98 / 100 ;
    }
    grs = basic + hra + da ;
    printf ( "gross salary = Rs. %f ", grs ) ;
}
```

# Class Exercise

An integer is input through keyboard. Write a program to determine if the number is even or odd:

The solution will be provided in the next class



# Nested if-else

It is perfectly all right if we write an entire if-else construct within either the body of the if statement or the body of an else statement.

This is called 'nesting' of ifs.

There is no limit on how deeply the ifs and the elses can be nested.

# Nested if-else

```
/* A quick demo of nested if-else */
```

```
main( )
```

```
{
```

```
    int i ;
```

```
    printf ( "Please Enter a numbar 1 or 2 " ) ;
```

```
    scanf ( "%d", &i ) ;
```

```
    if ( i == 1 )
```

```
        printf ( "You entered 1 !" ) ;
```

```
    else
```

```
    {
```

```
        if ( i == 2 )
```

```
            printf ( "You entered 2" ) ;
```

```
        else
```

```
            printf ( "you entered neither 1 nor 2 !" ) ;
```

```
    }
```

```
}
```

# Forms of if

a) if ( condition )

do this ;

(b) if ( condition )

{

do this ;

and this ;

}

(c) if ( condition )

✓ do this ;

else

✓ do this ;

(d) if ( condition )

{

do this ;

and this ;

}

else

{

do this ;

and this ;

}

(e) if ( condition )

do this ;

else

{

if ( condition )

do this ;

else

{

do this ;

and this ;

}

}

(f) if ( condition )

{

if ( condition )

do this ;

else

{

do this ;

and this ;

}

}

else

do this ;

# Logical Operators

C allows usage of three logical operators, namely, `&&`, `||` and `!`

These are 'AND' 'OR' and 'NOT' respectively

# Example

The marks obtained by a student in 5 different subjects are input through the keyboard. The student gets a division as per the following rules:

Percentage above or equal to 60 - First division

Percentage between 50 and 59 - Second division

Percentage between 40 and 49 - Third division

Percentage less than 40 - Fail

Write a program to calculate the division obtained by the student.

```

Void main( )
{
    int m1, m2, m3, m4, m5, per ;
    printf ( "Enter marks in five subjects " ) ;
    scanf ( "%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5 ) ;
    per = ( m1 + m2 + m3 + m4 + m5 ) / 5 ;
    if ( per >= 60 )
        printf ( "First division " ) ;
    else
    {
        if ( per >= 50 )
            printf ( "Second division" ) ;
        else
        {
            if ( per >= 40 )
                printf ( "Third division" ) ;
            else
                printf ( "Fail" ) ;
        }
    }
}

```

# Drawbacks

The program uses nested if-elses. This leads to three disadvantages:

1. As the number of conditions go on increasing the level of indentation also goes on increasing. As a result the whole program creeps to the right.
2. Care needs to be exercised to match the corresponding ifs and elses.
3. Care needs to be exercised to match the corresponding pair of braces.

All these three problems can be eliminated by usage of 'Logical operators'.

The following program illustrates this.

```
main( )
```

```
{
```

```
    int m1, m2, m3, m4, m5, per ;
```

```
    printf ( "Enter marks in five subjects " ) ;
```

```
    scanf ( "%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5 ) ;
```

```
    per = ( m1 + m2 + m3 + m4 + m5 ) / 5 ;
```

```
    if ( per >= 60 )
```

```
        printf ( "First division" ) ;
```

```
    if ( ( per >= 50 ) && ( per < 60 ) )
```

```
        printf ( "Second division" ) ;
```

```
    if ( ( per >= 40 ) && ( per < 50 ) )
```

```
        printf ( "Third division" ) ;
```

```
    if ( per < 40 )
```

```
        printf ( "Fail" ) ;
```

```
}
```



# Benefits of Logical Operators

Two distinct advantages can be cited in favor of this program:

1. The matching of the ifs with their corresponding elses gets avoided, since there are no elses in this program.
2. Despite of using several conditions, the program doesn't creep to the right

# The else-if Clause 1

```
void main( )
{
    int m1, m2, m3, m4, m5,per=5;
    clrscr();
    printf("Enter marks in five subjects ");
    scanf("%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5);
    per = ( m1+ m2 + m3 + m4+ m5 ) / per ;
    printf("percentage=%d\n",per);
    if ( per >= 60 )
        printf ( "First division" ) ;
    else if ( per >= 50 )
        printf ( "Second division" ) ;
    else if ( per >= 40 )
        printf ( "Third division" ) ;
    else
        printf ( "fail" ) ;
}
```

# The else-if Clause 2

This program reduces the indentation of the statements.  
In this case every else is associated with its previous if.  
The last else goes to work only if all the conditions fail  
The else if clause is just a way of rearranging the else with the if that follows it.

# The else-if Clause 3

```
if ( per >= 60 )
    printf ( "First division " ) ;
else
{
    if ( per >= 50 )
        printf ( "Second division" ) ;
}
if ( per >= 60 )
    printf ( "First division " ) ;
else if ( per >= 50 )
    printf ( "Second division");
```

# The ! Operator

The NOT operator is often used to reverse the logical value of a single variable, as in the expression

if ( ! flag )

This is another way of saying

if ( flag == 0 )

Does the NOT operator sound confusing? Avoid it if you want, as the same thing can be achieved without using the NOT operator.

# The ! Operator

This operator reverses the result of the expression it operates on.

if the expression evaluates to a non-zero value, then applying ! operator to it results into a 0


Vice versa, if the expression evaluates to zero then on applying ! operator to it makes it 1, a non-zero value.

- ! ( y < 10 )

We can express the same condition as ( y >= 10 )

# Hierarchy of Operators

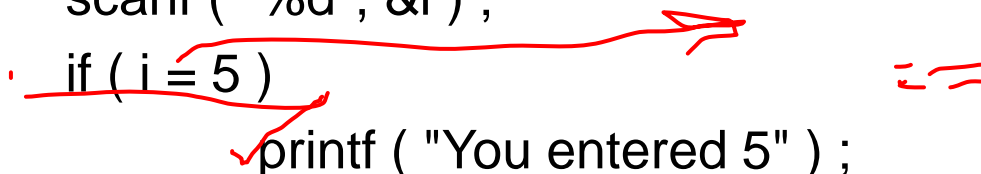
## Operators      Type



!	Logical NOT
* / %	Arithmetic and modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	Logical AND
	Logical OR
=	Assignment

# Hard to debug errors 1

```
void main( )  
{  
    int i ;  
    printf ( "Enter value of i " ) ;  
    scanf ( "%d", &i ) ;  
    if ( i = 5 )  
        printf ( "You entered 5" ) ;  
    else  
        printf ( "You entered something other than 5" ) ;  
}
```



## Output

Enter value of i 200

You entered 5

Enter value of i ~~9999~~

You entered 5



# Hard to debug errors 2

```
void main( )  
{  
    int i ;  
    printf ( "Enter value of i " ) ;  
    scanf ( "%d", &i ) ;  
    if ( i == 5 ) ;  
        printf ( "You entered 5" ) ;  
}  
    if ( i == 5 )  
        printf ( "You entered 5" ) ;
```

*Handwritten red annotations:*

- A red semicolon and "NVL" are written above the first closing brace of the main function.
- A red circle is drawn around the semicolon at the end of the first `if` statement.
- Red curly braces and arrows highlight the difference in indentation between the two `if` statements, illustrating how the first one is a null statement.

# Working of all three Logical Operators

Operands		Results			
x	y	!x	!y	x && y	x    y
0	0	1	1	0	0
0	non-zero	1	0	0	0
non-zero	0	0	1	0	1
non-zero	non-zero	0	0	1	1

# The Conditional Operators 1

conditional operators ? and : take 3 arguments  
they form a kind of foreshortened if-then-else  
e.g.

```
int x, y ;  
scanf ( "%d", &x ) ;  
y = ( x > 5 ? 3 : 4 ) ;
```

```
|  
| Its equivalent if statement is  
| if ( x > 5 )  
| y = 3 ;  
| else  
| y = 4 ;  
|
```

# The Conditional Operators 2

It's not necessary that the conditional operators should be used only in arithmetic statements. This is illustrated in the following examples:

Ex.:

```
int i ;
```

```
scanf ( "%d", &i ) ;
```

```
( i == 1 ? printf ( "one" ) : printf ( "not 1" ) ) ;
```

Ex.:

```
char a = 'z' ;
```

```
printf ( "%c" , ( a >123 ? a : '!' ) ) ;   output: !
```

# The Conditional Operators 3

The conditional operators can be nested as shown below:

```
int big, a=3, b=2, c=1 ;  
big = ( a > b ? ( a > c ? 3: 4 ) : ( b > c ? 6: 8 ) ) ;  
printf("%d\n",big);
```

Output: 3

```
int big, a=1, b=2, c=3 ;  
big = ( a > b ? ( a > c ? 3: 4 ) : ( b > c ? 6: 8 ) ) ;  
printf("%d\n",big);
```

Output: 8

# The Conditional Operators 4

```
int big, g=5,a=3, b=2, c=1 ;  
a > b ? g = a : g = b ;
```

This will give you an error 'Lvalue Required'. The error can be overcome by enclosing the statement in the : part within a pair of parenthesis. This is shown below:

```
int big, g=5,a=3, b=2, c=1 ;  
a > b ? g = a : (g = b );  
printf("%d\n",g);
```

Output: 3

# The Conditional Operators 5

The limitation of the conditional operators is that after the ? or after the : only one C statement can occur

In serious C programming conditional operators are not as frequently used as the if-else.

# Summary

- There are three ways for taking decisions in a program. First way is to use the if-else statement, second way is to use the conditional operators and third way is to use the switch statement.
- The default scope of the if statement is only the next statement. So, to execute more than one statement they must be written in a pair of braces.
- 
- An if block need not always be associated with an else block. However, an else block is always associated with an if statement.



# Summary

- If the outcome of an if-else ladder is only one of two answers then the ladder should be replaced either with an else-if clause or by logical operators.
- && and || are binary operators, whereas, ! is a unary operator.
- In C every test expression is evaluated in terms of zero and non-zero values. A zero value is considered to be false and a non-zero value is considered to be true.
- Assignment statements used with conditional operators must be enclosed within a pair of parenthesis.