Conditional Statements

By: Atiya Jokhio

We will learn:

Introduction to conditional statements

- If structure
- If –else structure
- If-else-if structure
- Nested If structure
- Switch statements

Introduction to conditional statements

- Conditional statements help you to make a decision based on certain conditions.
- These conditions are specified by a set of conditional statements having boolean expressions which are evaluated to a boolean value true or false.

If structure

- The single if statement in C language is used to execute the code if a condition is true.
- It is also called one-way selection statement.

Syntax:

```
if(expression)
{
 //code to be executed
}
```

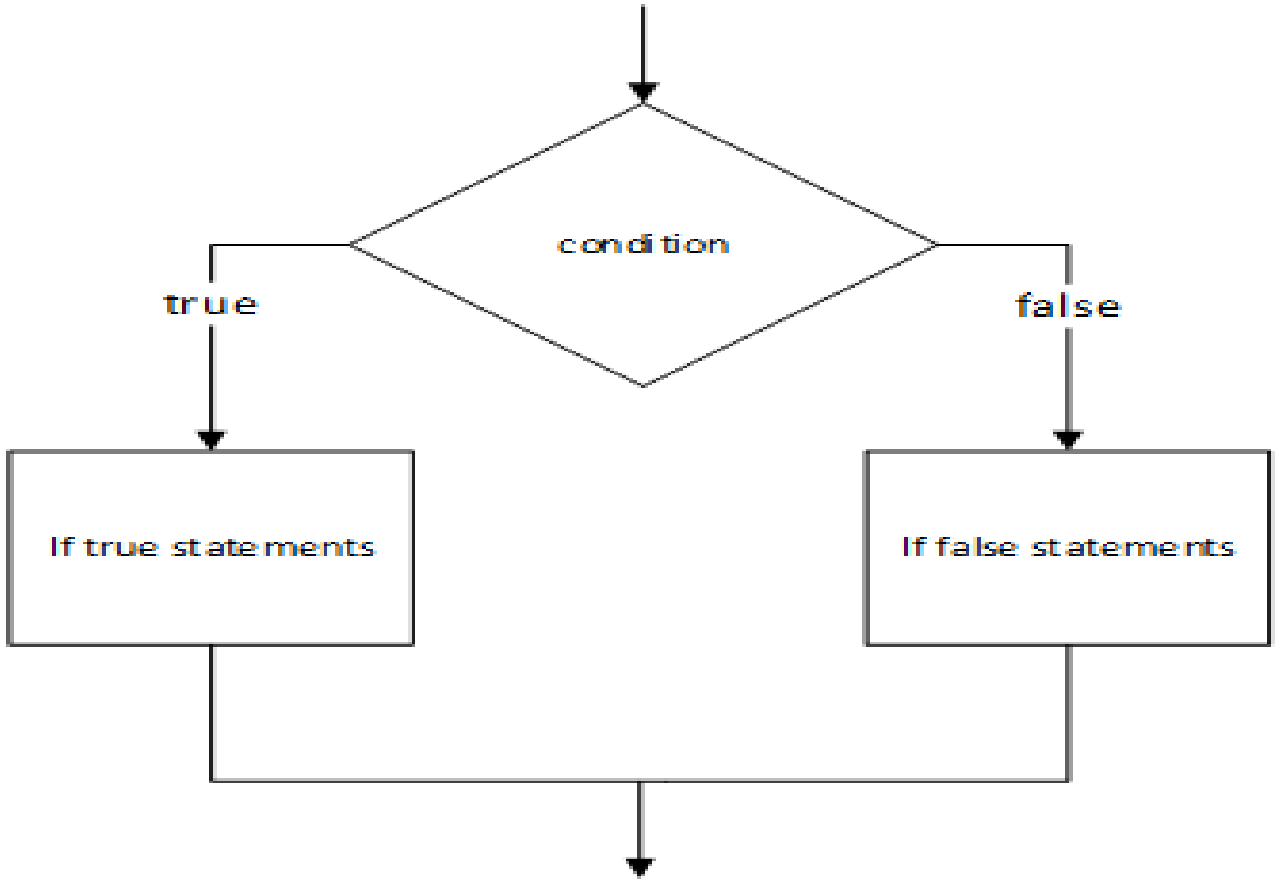
If structure [Cont.]

- As a general rule, we express a condition using C's 'relational' operators. The relational operators allow us to compare two values to see whether they are equal to each other, unequal, or whether one is greater than the other. Here's how they look and how they are evaluated in C.

this expression	is true if
x == y	x is equal to y
x != y	x is not equal to y
x < y	x is less than y
x > y	x is greater than y
x <= y	x is less than or equal to y
x >= y	x is greater than or equal to y

If structure [Cont.]

- Flowchart



If structure [Cont.]

- Example:

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 10;

   /* check the boolean condition using if statement */

   if( a < 20 ) {
      /* if condition is true then print the following */
      printf("a is less than 20\n" );
   }

   printf("value of a is : %d\n", a);

   return 0;
}
```

If structure [Cont.]

- TASK:

  While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses.

If structure [Cont.]

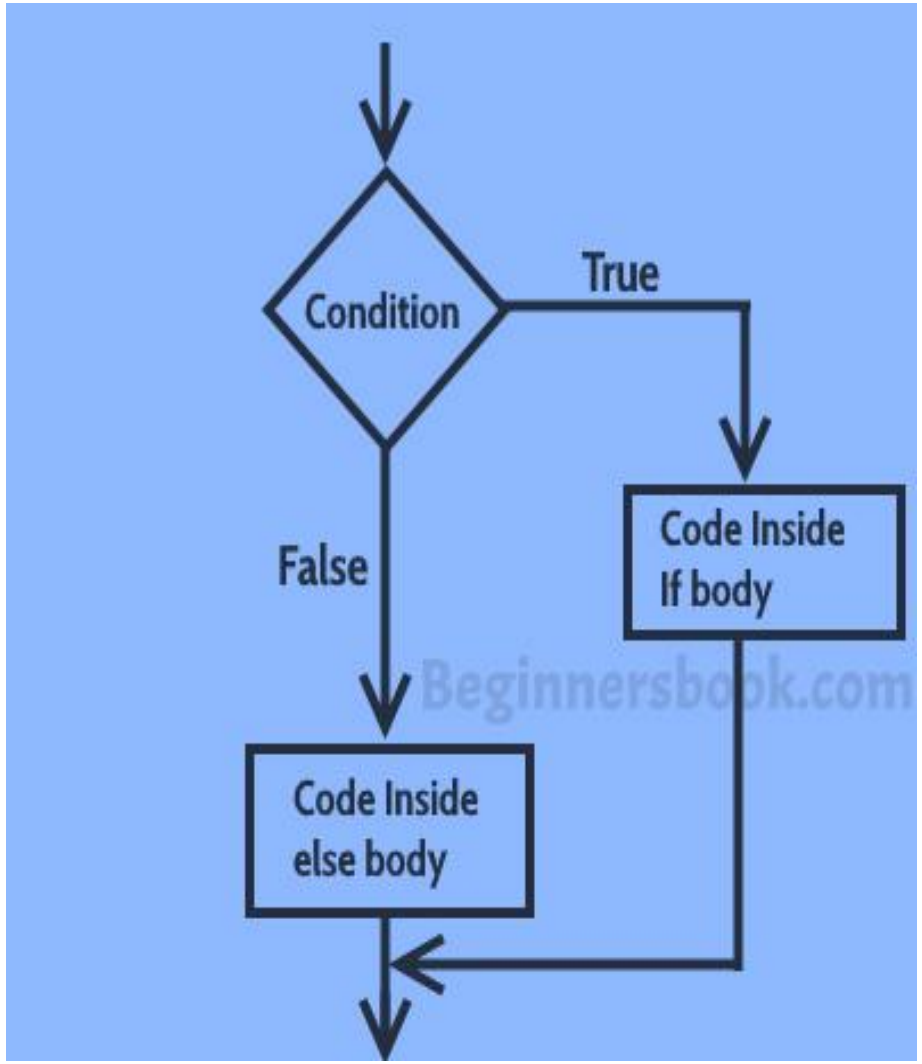
```
/* Calculation of total expenses*/
main( )
{
int qty, dis = 0 ;
float rate, tot ;
printf ( "Enter quantity and rate" ) ;
scanf ( "%d %f", &qty, &rate) ;
if ( qty > 1000 )
dis = 10 ;
tot = ( qty * rate ) - ( qty * rate * dis / 100 ) ;
printf ( "Total expenses = Rs. %f", tot ) ;
}
```

If else structure

- The decision logic structure uses the If/Then/Else instruction. It tells the computer that if a condition is true, then execute a set of instructions, or Else execute another set of instructions.
- The Else part is optional, as there is not always a set of instructions if the conditions are false.
- **Syntax:**

```
if(condition) {
   // Statements inside body of if
}
else {
   //Statements inside body of else
}
```

If else structure [Cont.]



If else structure [Cont.]

- Example:

Write a program in which user is asked to enter the age and based on the input, the if..else statement checks whether the entered age is greater than or equal to 18. If this condition meet then display message “You are eligible for voting”, however if the condition doesn’t meet then display a different message “You are not eligible for voting”.

If else structure [Cont.]

```c
#include <stdio.h>
int main()
{
    int age;
    printf("Enter your age:");
    scanf("%d",&age);
    if(age >=18)
    {
          /* This statement will only execute if the
           * above condition (age>=18) returns true
           */
          printf("You are eligible for voting");
    }
    else
    {
          /* This statement will only execute if the
           * condition specified in the "if" returns false.
           */
          printf("You are not eligible for voting");
    }
    return 0;
}
```

If else structure [Cont.]

TASK:

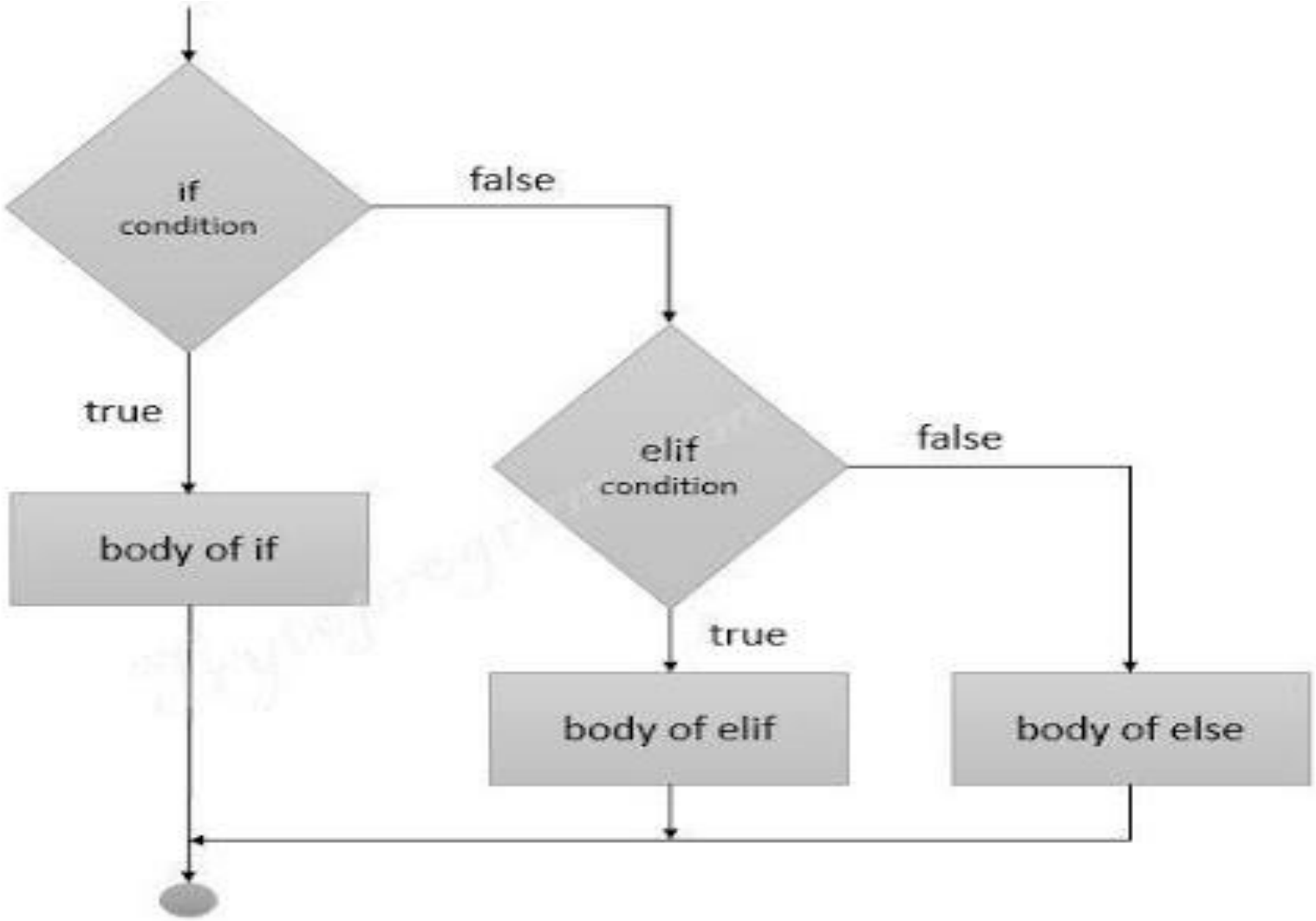
If his basic salary is less than Rs. 1500, then Allowance = 10% of basic salary and House Rent Allowance= 90% of basic salary.
If his salary is either equal to or above Rs. 1500, then Allowance = 500 and House Rent Allowance= 98% of basic salary.
If the employee's salary is input through the keyboard write a program to find his gross salary and also design flowchart.

If else if structure

- Multiple conditions can be written by making several else-if clauses. Once a condition is true, control will never go to other else-if conditions. An else clause can be added after else if statements.
- **Syntax:**

```
if (expression1)
      statement 1;
else if (expression 2)
      statement 2;
      ...
else
      statement n;
```

If else if structure [Cont.]



If else if structure [Cont.]

Example: Variable comparisons

```
#include <stdio.h>
int main()
{
int var1, var2;
printf("Input the value of var1:");
scanf("%d", &var1);
printf("Input the value of var2:");
scanf("%d",&var2);

      if (var1 !=var2)
      {
      printf("var1 is not equal to var2\n");
      }
      else if (var1 > var2)
      {
       printf("var1 is greater than var2\n");
      }
```

```
else if (var2 > var1)
{
printf("var2 is greater than var1\n");
}
else
{
printf("var1 is equal to var2\n");
}
return 0;
}
```

If else if structure [Cont.]

TASK:

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 and also draw flowchart.

If else if structure [Cont.]

```
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 else if structure [Cont.]

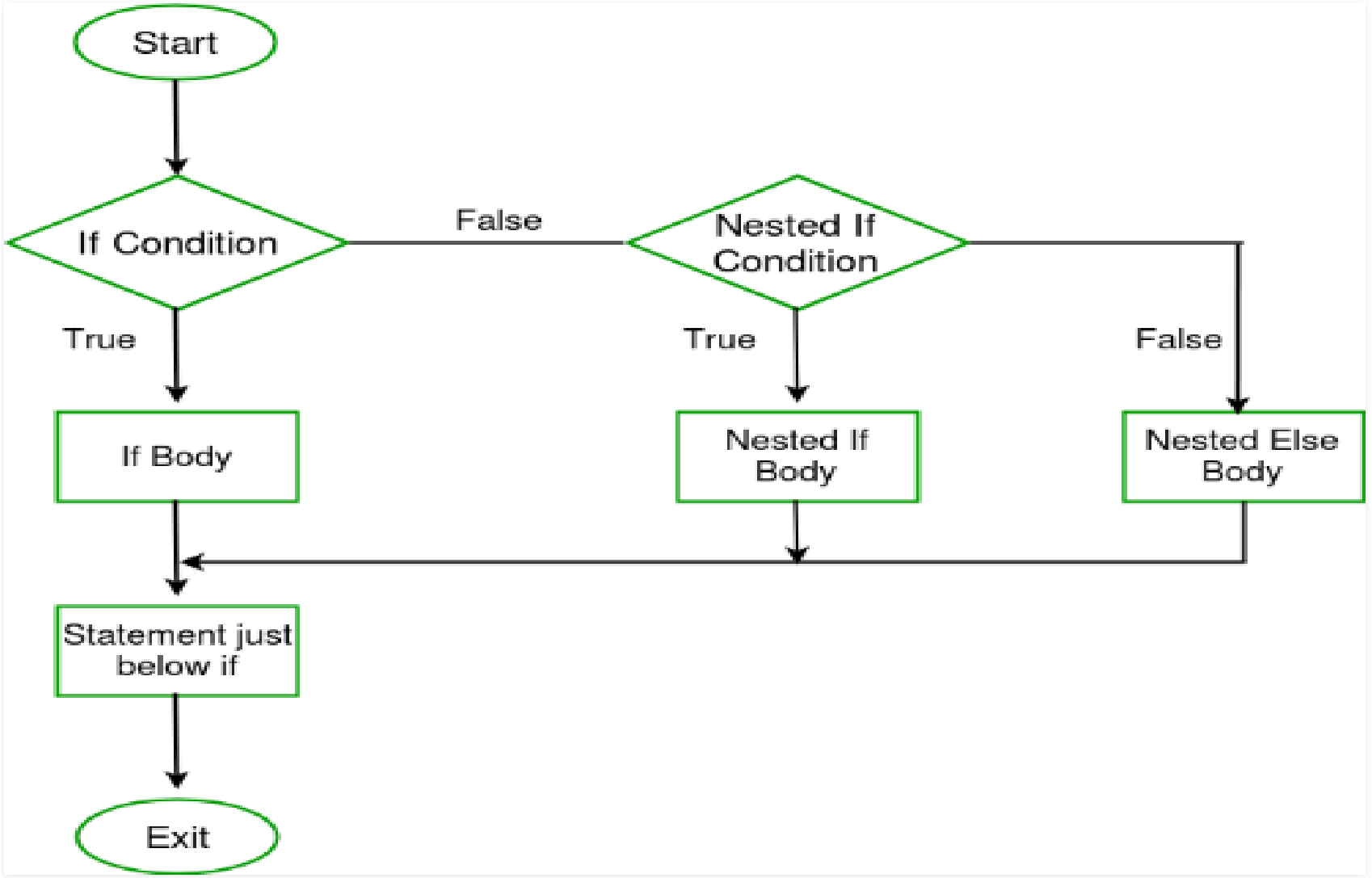
```
else if ( per >= 50 )
{
printf ( "Second division" ) ;
}
else if ( per >= 40 )
{
printf ( "Third division" ) ;
}
else
printf ( "Fail" ) ;
}
}
}
```

Nested if structure

- A nested if in C is an if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement.
- Syntax:

```
if (condition1)
{
    // Executes when condition1 is true
    if (condition2)
    {
        // Executes when condition2 is true
    }
}
```

Nested if structure [Cont.]



Nested if structure [Cont.]

Example:

```c
// C program to illustrate nested-if statement
#include <stdio.h>

int main() {
    int i = 10;

    if (i == 10)
    {
        // First if statement
        if (i < 15)
            printf("i is smaller than 15\n");

        // Nested - if statement
        // Will only be executed if statement above
        // is true
        if (i < 12)
            printf("i is smaller than 12 too\n");
        else
            printf("i is greater than 15");
    }

    return 0;
}
```

Logical Operators

- C allows usage of three logical operators, namely, &&, || and !.
- These are to be read as ‘AND’ ‘OR’ and ‘NOT’ respectively.

- Most obviously, two of them are composed of double symbols: ||and &&.

- The first two operators, &&and ||, allow two or more conditions to be combined in an if statement.

- The third logical operator is the NOT operator, written as !.

- This operator reverses the result of the expression it operates on. For example, 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.

Logical Operators [Cont.]

Operators	Example/Description
&& (logical AND)	(x>5)&&(y<5) It returns true when both conditions are true
|| (logical OR)	(x>=10)||(y>=10) It returns true when at-least one of the condition is true
! (logical NOT)	!((x>5)&&(y<5)) It reverses the state of the operand "((x>5) && (y<5))" If "((x>5) && (y<5))" is true, logical NOT operator makes it false

Logical Operators [Cont.]

```c
// C program to demonstrate working of logical operators
#include <stdio.h>

int main()
{
    int a = 10, b = 4, c = 10, d = 20;

    // logical operators

    // logical AND example
    if (a > b && c == d)
        printf("a is greater than b AND c is equal to d\n");
    else
        printf("AND condition not satisfied\n");

    // logical AND example
    if (a > b || c == d)
        printf("a is greater than b OR c is equal to d\n");
    else
        printf("Neither a is greater than b nor c is equal "
               " to d\n");

    // logical NOT example
    if (!a)
        printf("a is zero\n");
    else
        printf("a is not zero");

    return 0;
}
```

Logical Operators [Cont.]

Output:

```
AND condition not satisfied
a is greater than b OR c is equal to d
a is not zero
```

Logical Operators [Cont.]

TASK:
Write a C program by Using conditional operators to determine:

(1) Whether the character entered through the keyboard is a lower case alphabet or not.

(2) Whether a character entered through the keyboard is a special symbol or not.

Conditional Operators

- C provides the conditional operator (? and :), which is closely related to the if…else statement.
- The conditional operator is C's only ternary operator—it takes three operands.

These together with the conditional operator form a conditional expression.

- The first operand is a condition.
- The second operand is the value for the entire conditional expression if the condition is true
- and the third operand is the value for the entire conditional expression if the condition is false.

Example:

```
int i ;
scanf ( "%d", &i ) ;
( i == 1 ? printf ( “Hello" ) : printf ( “Bye" ) ) ;
```

Conditional Operators

- TASK:

Write a program to find maximum between three numbers using conditional operator

Switch Case Structure

- A switch statement tests the value of a variable and compares it with multiple cases. Once the case match is found, a block of statements associated with that particular case is executed.

- Each case in a block of a switch has a different name/number which is referred to as an identifier. The value provided by the user is compared with all the cases inside the switch block until the match is found.

Switch Case Structure [Cont.]

Syntax

```
switch( expression )
{
        case value-1:
                        Block-1;
                        Break;
        case value-2:
                        Block-2;
                        Break;
        case value-n:
                        Block-n;
                        Break;
        default:
                        Block-1;
                        Break;
}
Statement-x;
```

Switch Case Structure [Cont.]

- The expression can be integer expression or a character expression.
- Value-1, 2, n are case labels which are used to identify each case individually.
- Case labels always end with a colon (:). Each of these cases is associated with a block.
- A block is nothing but multiple statements which are grouped for a particular case.
- Whenever the switch is executed, the value of test-expression is compared with all the cases which we have defined inside the switch.

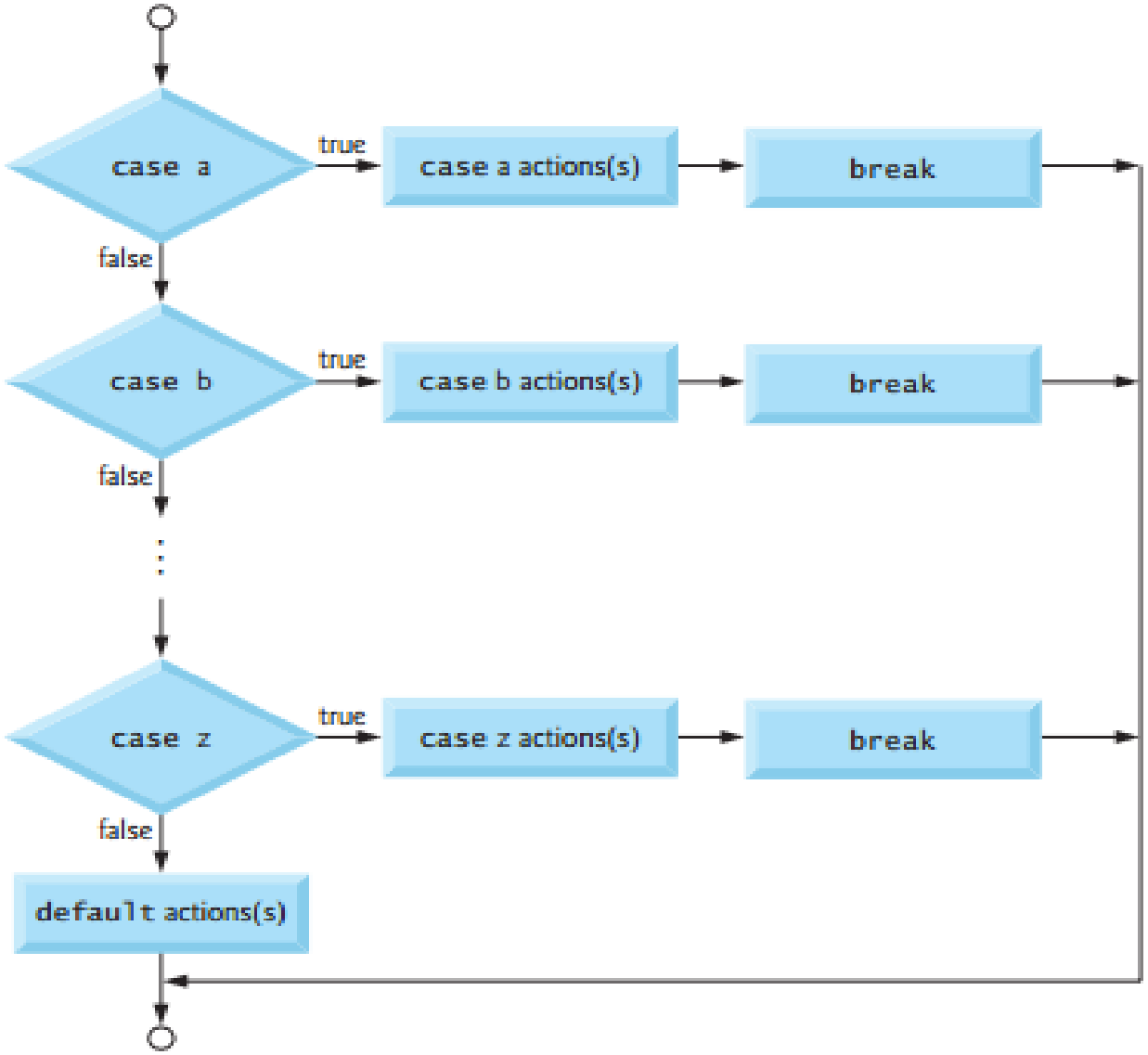
Switch Case Structure [Cont.]

- The break keyword in each case indicates the end of a particular case. If we do not put the break in each case then even though the specific case is executed, the switch will continue to execute all the cases until the end is reached. This should not happen; hence we always have to put break keyword in each case. Break will terminate the case once it is executed and the control will fall out of the switch.

- The default case is an optional one. Whenever the value of test-expression is not matched with any of the cases inside the switch, then the default will be executed.

- Otherwise, it is not necessary to write default in the switch.

Switch Case Structure [Cont.]



Switch Case Structure [Cont.]

Example:

```c
#include <stdio.h>
int main() {
int language = 10;
  switch (language) {
  case 1:
    printf("C#\n");
    break;
  case 2:
    printf("C\n");
    break;
  case 3:
    printf("C++\n");
    break;
  default:
    printf("Other programming language\n");}}
```

```c
#include <stdio.h>
int main() {
int number=5;
switch (number) {
  case 1:
  case 2:
  case 3:
    printf("One, Two, or Three.\n");
    break;
  case 4:
  case 5:
  case 6:
    printf("Four, Five, or Six.\n");
    break;
  default:
    printf("Greater than Six.\n");}}
```

Nested Switch

- In C, we can have an inner switch embedded in an outer switch. Also, the case constants of the inner and outer switch may have common values and without any conflicts.

Nested Switch [Cont.]

```
#include <stdio.h>
int main() {
        int ID = 500;
        int password = 000;
        printf("Plese Enter Your ID:\n ");
        scanf("%d", & ID);
        switch (ID) {
            case 500:
                printf("Enter your password:\n ");
                scanf("%d", & password);
                switch (password) {
                    case 000:
                        printf("Welcome Dear Programmer\n");
                        break;
                    default:
                        printf("incorrect password");
                        break;
                }
                break;
            default:
                printf("incorrect ID");
                break;
        }
}
```

Nested Switch

✓ **Why do we need a Switch case?**

- There is one potential problem with the if-else statement which is the complexity of the program increases whenever the number of alternative path increases. If you use multiple if-else constructs in the program, a program might become difficult to read and comprehend.
- Sometimes it may even confuse the developer who himself wrote the program.
- The solution to this problem is the switch statement.

✓ **TASK:**

Write a C program to print number of days in a month when user inputs the number of month using switch condition.

```c
/* C Program to Print Number of Days in a Month using Switch Condition */
#include <stdio.h>
int main()
{
  int month;
  printf(" Please Enter the Month Number 1 to 12 (Consider 1 = January, and 12 = December) :  ");
  scanf("%d", &month);

  switch(month )
  {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        printf("\n 31 Days in this Month");
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        printf("\n 30 Days in this Month");
        break;
    case 2:
        printf("\n Either 28 or 29 Days in this Month");
    default:
        printf("\n Please enter Valid Number between 1 to 12");
  }
  return 0;
}
```