Fundamentals of Programming CCS1063/CSE1062

Lecture 5 –Operators Part 2 , Conditions and Loops

Professor Noel Fernando

Increment & Decrement Operators

- C programming has two increment and decrement operators to change the value of an operand (constant or variable) by 1.
  - Increment ++ increases the value by 1
  - Decrement -- decreases the value by 1
- These two operators are unary operators, meaning they only operate on a single operand.
- The operators ++ and -- can be used as prefix or postfix.

Increment & Decrement Operators

- **++ and -- operator as prefix and postfix**
- If you use the ++ operator as a prefix like:++var the value of var is incremented by 1; then it returns the value.
- If you use the ++ operator as a postfix like : var++, the original value of var is returned first; then var is incremented by 1.
- The -- operator works in a similar way to the ++ operator except -- decreases the value by 1.

Increment & Decrement Operators

```c
int main() {
    int a = 7;
    float b = 5.5;
    printf("++a = %d\n", ++a);
    printf("--b = %.2f\n", --b);
    printf("a++ = %d\n", a++);
    printf("b-- = %.2f\n", b--);
    printf("Final Values: a = %d, b = %.2f\n", a, b);
}
```

Increment & Decrement Operators

```c
int main() {
    int a = 7;
    float b = 5.5;
    printf("++a = %d\n", ++a);
    printf("--b = %.2f\n", --b);
    printf("a++ = %d\n", a++);
    printf("b-- = %.2f\n", b--);
    printf("Final Values: a = %d, b = %.2f\n", a, b);
}
```

```
++a = 8
--b = 4.50
a++ = 8
b-- = 4.50
Final Values: a = 9, b = 3.50
```

Assignment Operators

- An assignment operator is used for assigning a value to a variable.
  - The most common assignment operator is =

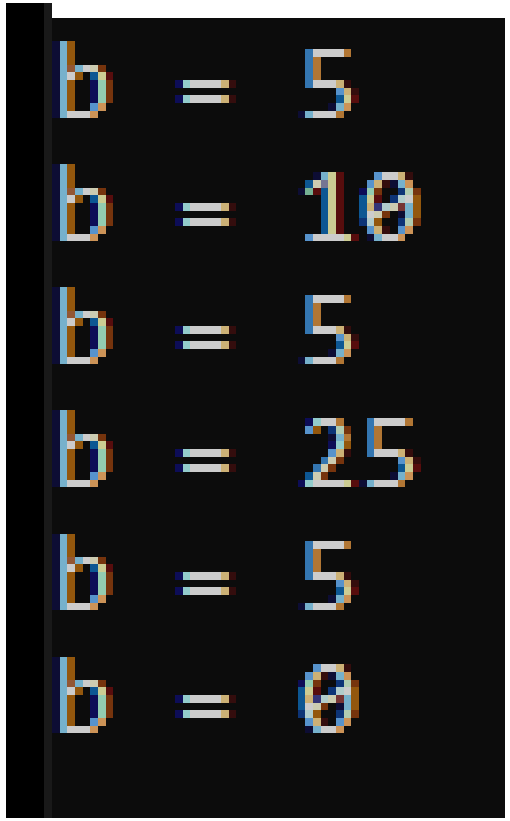
Operator	Example	Same as…
=	a = b	a = b
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b

Assignment Operators

```c
int main() {
    int a = 5, b;
    b = a;
    printf("b = %d\n", b);
    b += a;
    printf("b = %d\n", b);
    b -= a;
    printf("b = %d\n", b);
    b *= a;
    printf("b = %d\n", b);
    b /= a;
    printf("b = %d\n", b);
    b %= a;
    printf("b = %d\n", b);
}
```

Assignment Operators

```c
int main() {
    int a = 5, b;
    b = a;
    printf("b = %d\n", b);
    b += a;
    printf("b = %d\n", b);
    b -= a;
    printf("b = %d\n", b);
    b *= a;
    printf("b = %d\n", b);
    b /= a;
    printf("b = %d\n", b);
    b %= a;
    printf("b = %d\n", b);
}
```



Relational Operators

- A relational operator checks the relationship between two operands.
- If the relation is true, it returns 1; if the relation is false, it returns value 0
- Relational operators are used in decision making and loops.

Operator	Meaning	Example
==	Equal to	5 == 3 returns 0
>	Greater than	5 > 3 returns 1
<	Less than	5 < 3 returns 0
!=	Not equal to	5 != 3 returns 1
>=	Greater than or equal to	5 >= 3 returns 1
<=	Less than or equal to	5 <= 3 returns 0

Relational Operators

```c
int main() {

    int a = 5, b = 5, c = 10;

    printf("%d == %d = %d\n", a, b, a == b);
    printf("%d == %d = %d\n", a, c, a == c);

    printf("%d > %d = %d\n", a, b, a > b);
    printf("%d > %d = %d\n", a, c, a > c);

    printf("%d < %d = %d\n", a, b, a < b);
    printf("%d < %d = %d\n", a, c, a < c);

    printf("%d != %d = %d\n", a, b, a != b);
    printf("%d != %d = %d\n", a, c, a != c);

    printf("%d >= %d = %d\n", a, b, a >= b);
    printf("%d >= %d = %d\n", a, c, a >= c);

    printf("%d <= %d = %d\n", a, b, a <= b);
    printf("%d <= %d = %d\n", a, c, a <= c);

}
```

Relational Operators

```c
int main() {

    int a = 5, b = 5, c = 10;

    printf("%d == %d = %d\n", a, b, a == b);
    printf("%d == %d = %d\n", a, c, a == c);

    printf("%d > %d = %d\n", a, b, a > b);
    printf("%d > %d = %d\n", a, c, a > c);

    printf("%d < %d = %d\n", a, b, a < b);
    printf("%d < %d = %d\n", a, c, a < c);

    printf("%d != %d = %d\n", a, b, a != b);
    printf("%d != %d = %d\n", a, c, a != c);

    printf("%d >= %d = %d\n", a, b, a >= b);
    printf("%d >= %d = %d\n", a, c, a >= c);

    printf("%d <= %d = %d\n", a, b, a <= b);
    printf("%d <= %d = %d\n", a, c, a <= c);

}
```

```
5 == 5 = 1
5 == 10 = 0
5 > 5 = 0
5 > 10 = 0
5 < 5 = 0
5 < 10 = 1
5 != 5 = 0
5 != 10 = 1
5 >= 5 = 1
5 >= 10 = 0
5 <= 5 = 1
5 <= 10 = 1
```

Logical Operators

- An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND, true only if all operands are true	If c = 5 & d = 2, then, expression ((c==5)&&(d==5)) equals to 0
||	Logical OR, true only if either operand is true	If c = 5 & d = 2, then, expression ((c==5)&&(d==5)) equals to 1
!	Logical NOT, true only if the operand is 0	If c = 5 then, expression !(c==5) equals to 0

Logical Operators

```c
int main() {

    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) equals to %d\n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) equals to %d\n", result);

    result = (a == b) || (c > b);
    printf("(a == b) || (c > b) equals to %d\n", result);

    result = (a != b) || (c > b);
    printf("(a != b) || (c > b) equals to %d\n", result);

    result = !(a != b);
    printf("!(a != b) equals to %d\n", result);

    result = !(a == b);
    printf("!(a == b) equals to %d\n", result);
}
```

Logical Operators

```c
int main() {

    int a = 5, b = 5, c = 10, result;

    result = (a == b) && (c > b);
    printf("(a == b) && (c > b) equals to %d\n", result);

    result = (a == b) && (c < b);
    printf("(a == b) && (c < b) equals to %d\n", result);

    result = (a == b) || (c > b);
    printf("(a == b) || (c > b) equals to %d\n", result);

    result = (a != b) || (c > b);
    printf("(a != b) || (c > b) equals to %d\n", result);

    result = !(a != b);
    printf("!(a != b) equals to %d\n", result);

    result = !(a == b);
    printf("!(a == b) equals to %d\n", result);
}
```

```
(a == b) && (c > b) equals to 1
(a == b) && (c < b) equals to 0
(a == b) || (c > b) equals to 1
(a != b) || (c > b) equals to 1
!(a != b) equals to 1
!(a == b) equals to 0
```

Bitwise Operators

- During computation, mathematical operations like: addition, subtraction, Multiplication and division are converted to bit-level which makes processing faster and saves power.
- Bitwise operators are still used for those working on embedded devices that have memory limitations.
- Bitwise operators are used in C programming to perform bit-level operations.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

Basics of Bitwise Operations

Name	Symbol	Usage	What it does
Bitwise And	&	a &b	Returns 1 only if both the bits are 1
Bitwise Or	|	a | b	Returns 1 if one of the bits is 1
Bitwise Not	~	~a	Returns the complement of a bit
Bitwise Xor	^	a^b	Returns 0 if both the bits are same else 1
Bitwise Left shift	<<	a<<n	Shifts a towards left by n digits
Bitwise Right shift	>>	a>>n	Shifts a towards right by n digits

Bitwise AND (&)

- The output of bitwise AND is 1 if the corresponding bits of two operands are 1.
  - If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

```
12  =  00001100 (in Binary)
25  =  00011001 (in Binary)

Bit operation of 12 and 25

     00001100
&    00011001
     --------
     00001000  =  8 (in Decimal)
```

C code for Bitwise AND

```
#include <stdio.h>

int main() {

int a = 12, b = 25;

printf("Output = %d", a & b);
return 0;}
```

- Output is :
- Output = 8

Don’t get confused with Binary Addition

- **Examples of Binary Addition**
- **Example 1:** 10001 + 11101
- **Solution:**

```
          1
    1 0 0 0 1
(+) 1 1 1 0 1
———————————
  1 0 1 1 1 0
```

- **Rules of Binary Addition**
- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 =10

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	0 (where 1 s carried over)

Bitwise OR (|)

- The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1

```
12  =  00001100 (in Binary)
25  =  00011001 (in Binary)

Bit operation of 12 and 25

    00001100
  & 00011001
    ________

    00011101  =  29 (in Decimal)
```

C code for Bitwise OR

```
#include <stdio.h>
int main() {
int a = 12, b = 25;
printf("Output = %d", a | b);
return 0;}
```

Output is :

Output = 29

Bitwise XOR (^)

- The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite.

```
12  =  00001100 (in Binary)
25  =  00011001 (in Binary)

Bit operation of 12 and 25

    00001100
^   00011001
    --------
    00010101  =  21 (in Decimal)
```

- C code for Bitwise XOR

```
#include <stdio.h>
int main() {
 int a = 12, b = 25;
printf("Output = %d", a ^ b);
return 0;}
```

Output :

Output = 21

Bitwise complement (~)

- Bitwise compliment operator is an unary operator (works on only one operand) which changes 1 to 0 and 0 to 1.

```
12  =  00001100 (in Binary)

Bitwise complement of 12

   ~ 00001100
     ----------
     11110011  =  243 (in Decimal)
```

- Example :C code for Bitwise complement

```
#include <stdio.h>

int main() {

printf("Output = %d\n", ~35);
printf("Output = %d\n", ~-12);
return 0;}
```

Outputs:

Output = -36

Output = 11

Bitwise Operators

```c
int main() {

    int a = 12, b = 25;

    printf("Bitwise AND = %d\n", a&b);
    printf("Bitwise OR = %d\n", a|b);
    printf("Bitwise XOR = %d\n", a^b);
    printf("Complement of a = %d\n", ~a);
    printf("Complement of b = %d\n", ~b);

}
```

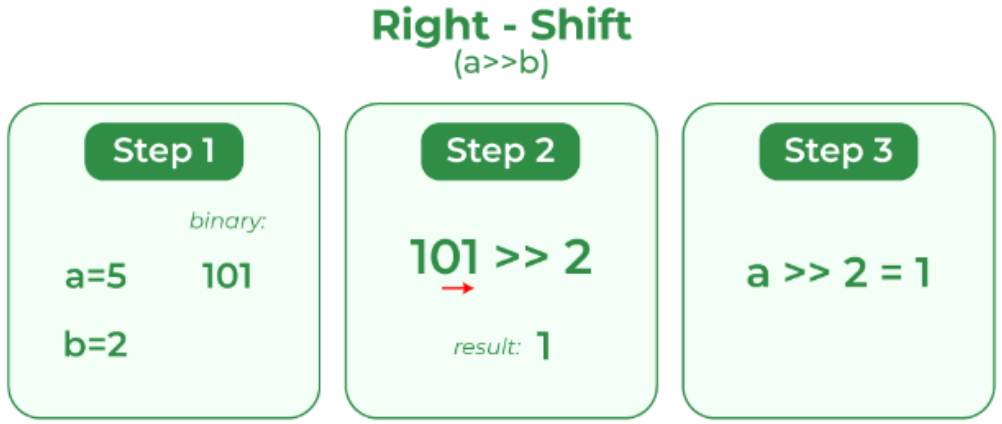
Bitwise Operators

```c
int main() {

    int a = 12, b = 25;

    printf("Bitwise AND = %d\n", a&b);
    printf("Bitwise OR = %d\n", a|b);
    printf("Bitwise XOR = %d\n", a^b);
    printf("Complement of a = %d\n", ~a);
    printf("Complement of b = %d\n", ~b);

}
```

```
Bitwise AND = 8
Bitwise OR = 29
Bitwise XOR = 21
Complement of a = -13
Complement of b = -26
```

Shift Operators

- Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

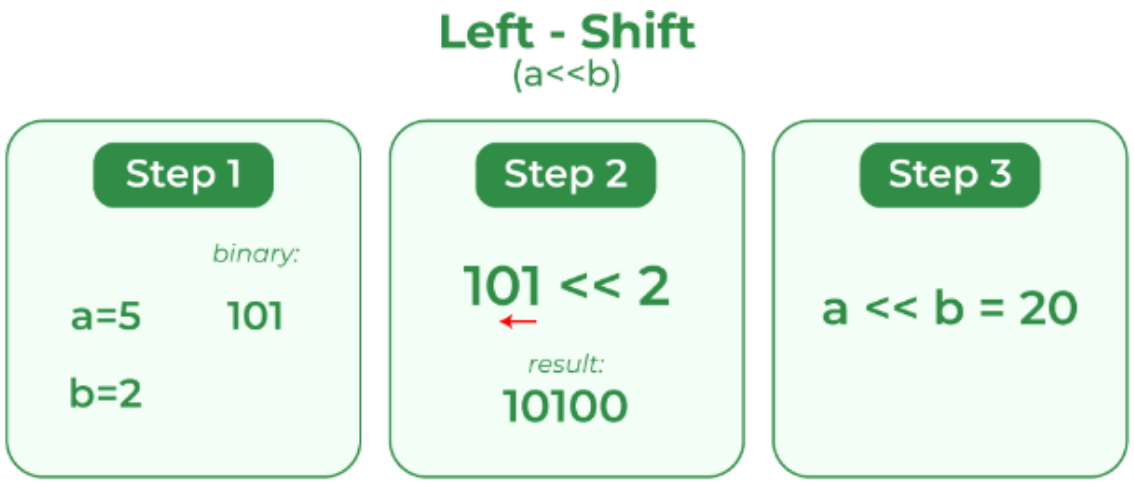


- Right shift operator

```
212 = 11010100 (In binary)
212>>2 = 00110101 (In binary) [Right shift by two bits]
212>>7 = 00000001 (In binary)
212>>8 = 00000000
212>>0 = 11010100 (No Shift)
```

Shift Operators

- Left shift operator shifts all bits towards left by a certain number of specified bits.
- The bit positions that have been vacated by the left shift operator are filled with **0**. The symbol of the left shift operator is <<
- Left shift operator




```
212 = 11010100 (In binary)
212<<1 = 110101000 (In binary) [Left shift by one bit]
212<<0 =11010100 (Shift by 0)
212<<4 = 110101000000 (In binary) =3392(In decimal)
```

Example : Shift Operators (Left & Right)

```c
#include <stdio.h>

int main() {

    int num=212, i;

    for (i = 0; i <= 2; ++i) {
        printf("Right shift by %d: %d\n", i, num >> i);
    }
    printf("\n");

    for (i = 0; i <= 2; ++i) {
        printf("Left shift by %d: %d\n", i, num << i);
    }

    return 0;
}
```

Output:

Right shift by 0: 212
Right shift by 1: 106
Right shift by 2: 53

Left shift by 0: 212
Left shift by 1: 424
Left shift by 2: 848

Other Operators

- Comma Operator
  - Comma operators are used to link related expressions together.
  - For example:

```
int x = 10, y = 5, z;
```

- The sizeof operator
  - The *sizeof* is an unary operator which returns the size of data (constant, variables, array, structure etc).

- C Ternary Operator (?:)

Exercise 1-Write down the output if one executes the following program.

```
#include <stdio.h>
int main()
{
unsigned char a = 5, b = 9;
printf("a = %d, b = %d\n", a, b);
printf("a&b = %d\n", a & b);
printf("a|b = %d\n", a | b);
printf("a^b = %d\n", a ^ b);
printf("~a = %d\n", a = ~a);
printf("b<<1 = %d\n", b << 1);
printf("b>>1 = %d\n", b >> 1);
return 0;
}
```

- **Output**

Exercise 1-Write down the output if one executes the following program.

```
#include <stdio.h>
int main()
{
unsigned char a = 5, b = 9;
printf("a = %d, b = %d\n", a, b);
printf("a&b = %d\n", a & b);
printf("a|b = %d\n", a | b);
printf("a^b = %d\n", a ^ b);
printf("~a = %d\n", a = ~a);
printf("b<<1 = %d\n", b << 1);
printf("b>>1 = %d\n", b >> 1);
return 0;
}
```

- **Output**

```
a = 5, b = 9
a & b = 1
a | b = 13
a ^ b = 12
~a = -6
b << 1 = 18
b >> 1 = 4
```

Exercise 2

a) Write a C program to swap two numbers without using temporary variable
b) Write a C program to swap two numbers using Bitwise operators.

Exercise 2

a)Write a C program to swap two numbers without using temporary variable

```
#include <stdio.h>
int main()
{
int x = 10, y = 5;
// Code to swap 'x' and 'y'
x = x * y; // x now becomes 50
y = x / y; // y becomes 10
x = x / y; // x becomes 5
printf("After Swapping: x = %d, y = %d", x, y);
return 0;
}
```

- **Output**

```
After Swapping: x =5, y=10
```

Exercise 2

(b) **Write a C program to swap two numbers using Bitwise operators.**

Most suitable operator is :Bitwise XOR

For example,

XOR of 10 (In Binary 1010) and

5 (In Binary 0101) is 1111, and

XOR of 7 (0111) and 5 (0101) is (0010)

C code for swapping

```c
#include <stdio.h>
int main()
{
    int x = 10, y = 5;
      // Code to swap 'x' (1010) and 'y' (0101)
    x = x ^ y; // x now becomes 15 (1111)
    y = x ^ y; // y becomes 10 (1010)
    x = x ^ y; // x becomes 5 (0101)

    printf("After Swapping: x = %d, y = %d", x, y);

    return 0;
}
```

Output:

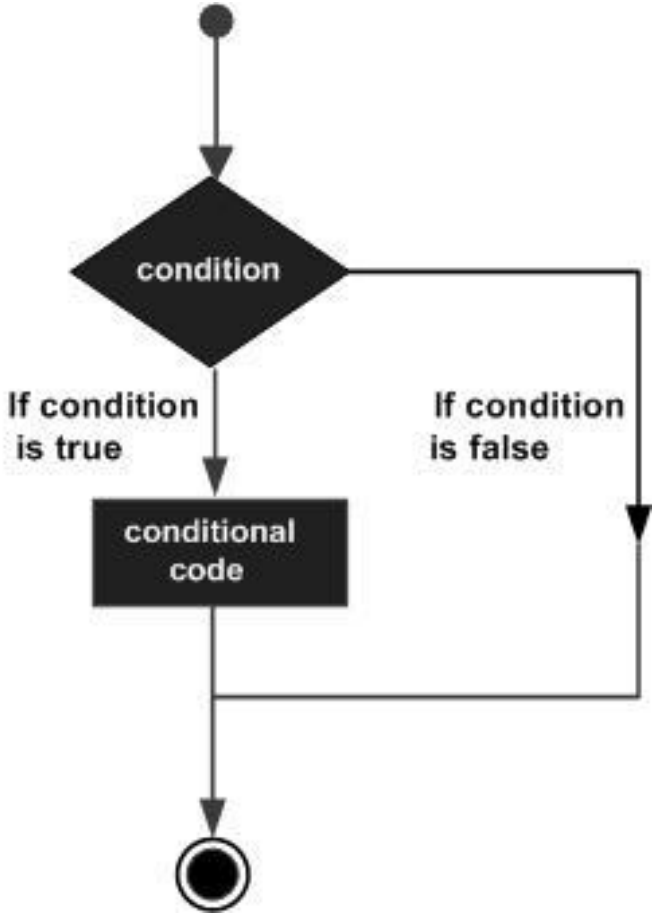
```
After Swapping: x =5, y=10
```

Conditional statements

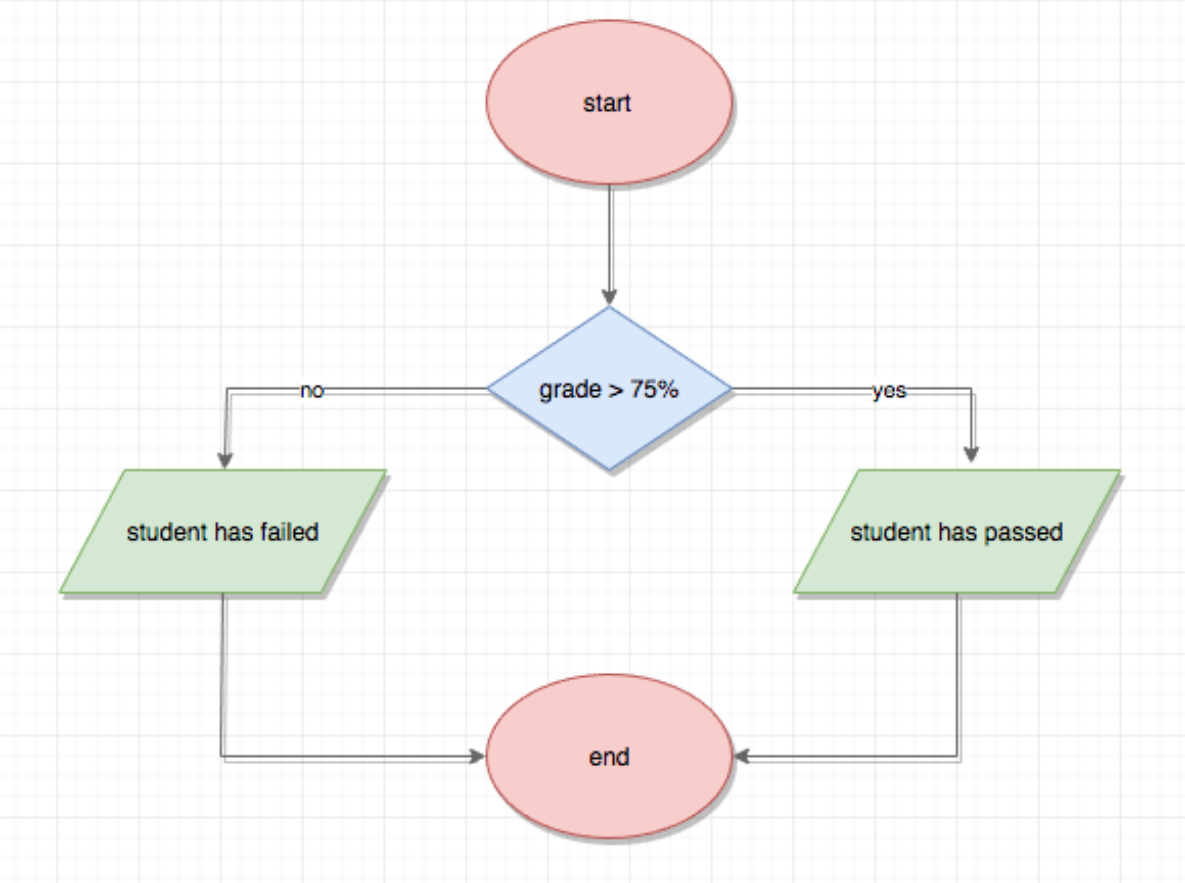
- Conditional statements in C are programming constructs that allow a program to execute different blocks of code based on whether a certain condition is true or false.
- The most common types of conditional statements in C are the if, else if, else statements and case statements.

Condition

- In programming, decision making is used to specify the order in which statements are executed.
- Show below is the general form of a typical decision making structure found in most of the programming languages.



Problem –
Work out whether the student passed or failed



Begin

Input mark

If (mark>=75)

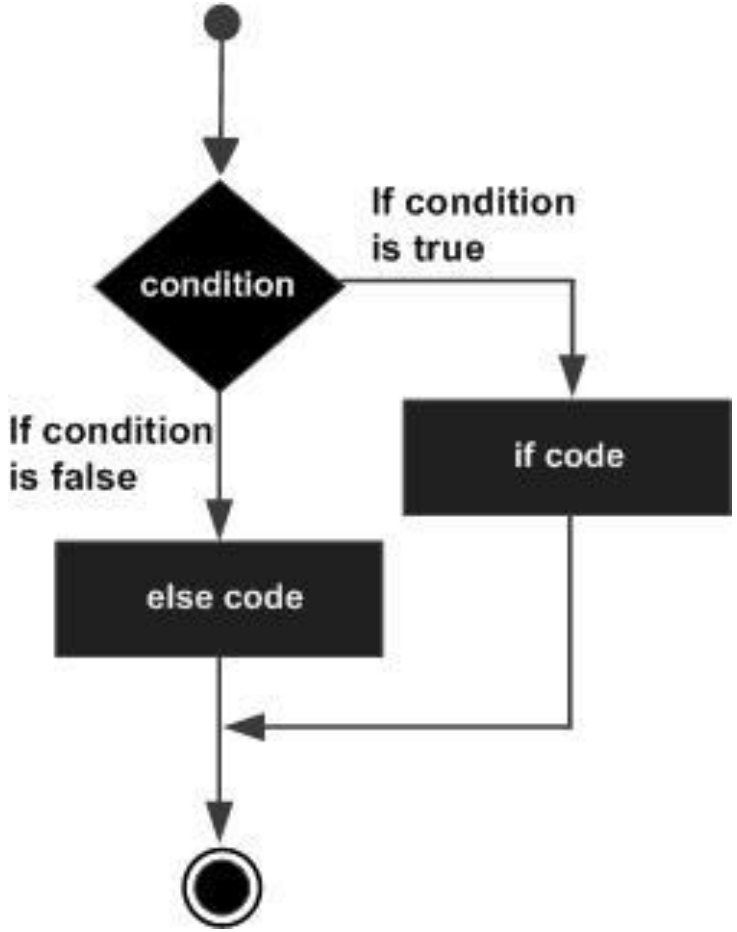
  print (“pass”)

Else

Print (“fail”)

end

Syntax - if then else statements




```
if(boolean_expression) {
   /* statement(s) will execute if the boolean expression is true */
} else {
   /* statement(s) will execute if the boolean expression is false */
}
```

Write a C program for Work out whether the student passed or failed

```
#include <stdio.h>
int main () {
int mark = 73;
if( mark >70 ) {
printf(“Pass\n" );
} else
printf(“Fail\n" );   }
printf("value of mark is : %d\n", mark);
return 0;}
```

- pass
- value of a is : 73

Question

- Write a C program to determine whether the given number is odd or Even
- E.g. (i) if input number is 11, output should be “Odd”
- (ii) if input number is 16, output should be “Even”

Odd & Even Numbers

```c
int main() {

    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    if(number%2==0) {
        printf("%d is a even number.", number);
    }
    else {
        printf("%d is an odd number.", number);
    }
}
```

What if you need to print done! after printing the massage?

nested if-else statement

- A ***nested if-else statement*** is an ***if statement*** inside another ***if statement***.

- The general syntax of nested if-else statement in C is as follows:

```
if (condition1) {
   /* code to be executed if conditio
n1 is true */
   if (condition2) {
      /* code to be executed if conditi
on2 is true */
   } else {
      /* code to be executed if conditi
on2 is false */
   }
} else {
   /* code to be executed if conditio
n1 is false */
}
```

Question

- Draw a flow chart and Write a C program that takes in a number and checks whether it is ***positive, negative***, or ***zero***.

Question

- Draw a flow chart and Write a C program that takes in a number and checks whether it is ***positive, negative***, or ***zero***.

```
#include <stdio.h>
  int main() {
   int num;
 printf("Enter a number: ");
scanf("%d", &num);
     if (num > 0) {
printf("%d is positive.\n", num);
   } else {
      if (num < 0) {
printf("%d is negative.\n", num);
      } else {
printf("%d is zero.\n", num);
      }
   }
     return 0;
}
```

Example: Grades

- Write a program to give the grade when you enter your mark of a subject.
- The grade of the mark is defined as follows.
  - 00 – 24: E
  - 25 – 34: D
  - 35 – 49: C
  - 50 – 69: B
  - 70 – 100: A

- 00 – 24: E
- 25 – 34: D
- 35 – 49: C
- 50 – 69: B
- 70 – 100: A

Example: Grades

- Write a program to give the grade when you enter your mark of a subject.
- The grade of the mark is defined as follows.
  - 00 – 24: E
  - 25 – 34: D
  - 35 – 49: C
  - 50 – 69: B
  - 70 – 100: A

```c
int main() {

    int mark;
    char grade;

    printf("Enter your mark: ");
    scanf("%d", &mark);

    if(mark>69) {
        grade = 'A';
    }
    else if(mark>49) {
        grade = 'B';
    }
    else if(mark>34) {
        grade = 'C';
    }
    else if(mark>24) {
        grade = 'D';
    }
    else {
        grade = 'E';
    }
    printf("your grade is %c", grade);
}
```

Question :

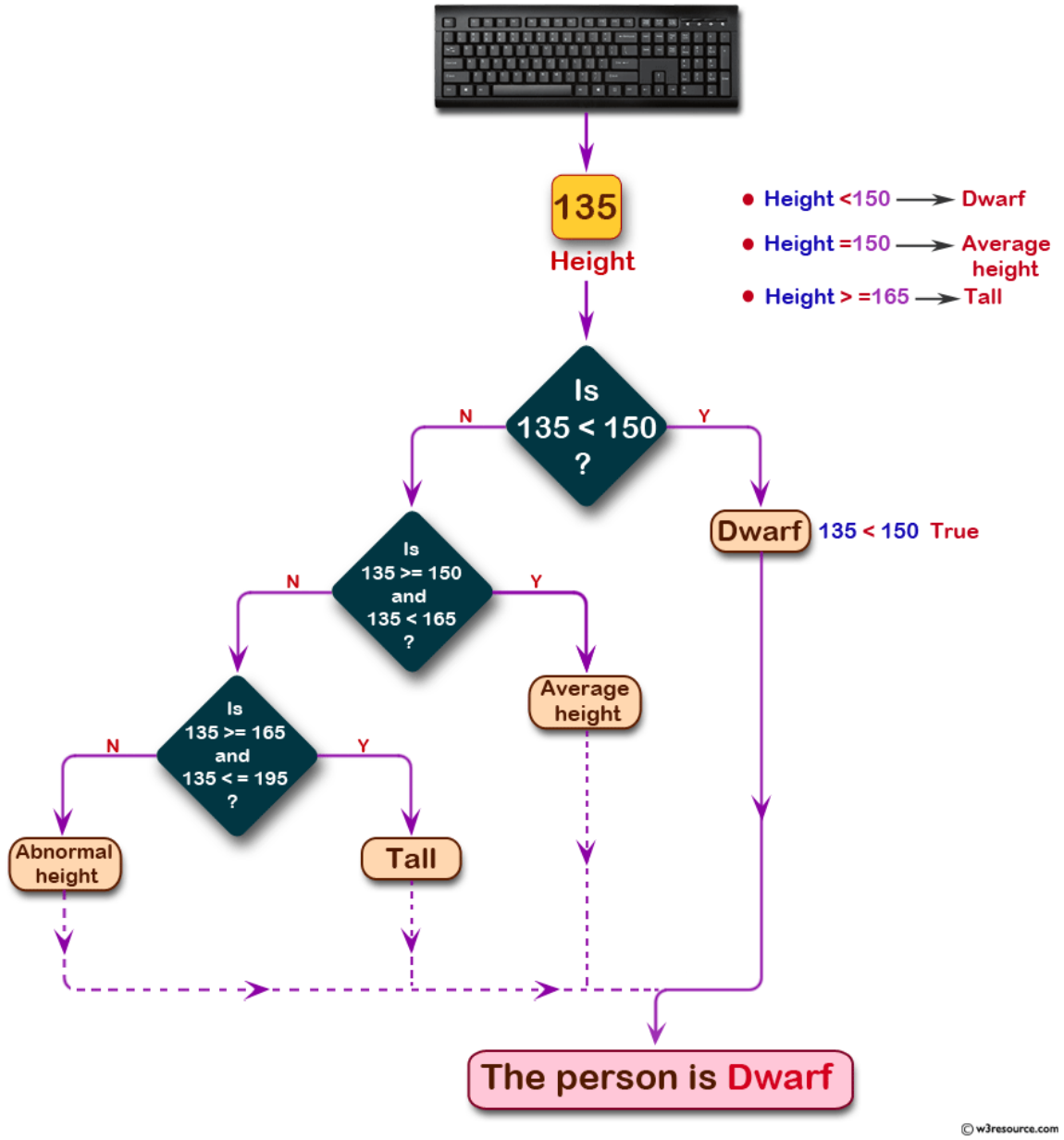
- Draw flow chart and Write a C program to accept the height of a person in centimeters and categorize the person according to their height.
- Categorization criteria is given below:

  - Height <150 ⟶ Dwarf
  - Height =150 ⟶ Average height
  - Height > =165 ⟶ Tall

Question :

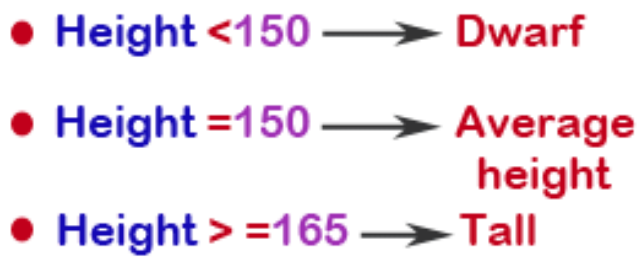
- Draw flow chart and Write a C program to accept the height of a person in centimeters and categorize the person according to their height.
- Categorization criteria is given below:

  - Height <150 ⟶ Dwarf
  - Height =150 ⟶ Average height
  - Height > =165 ⟶ Tall



Question :

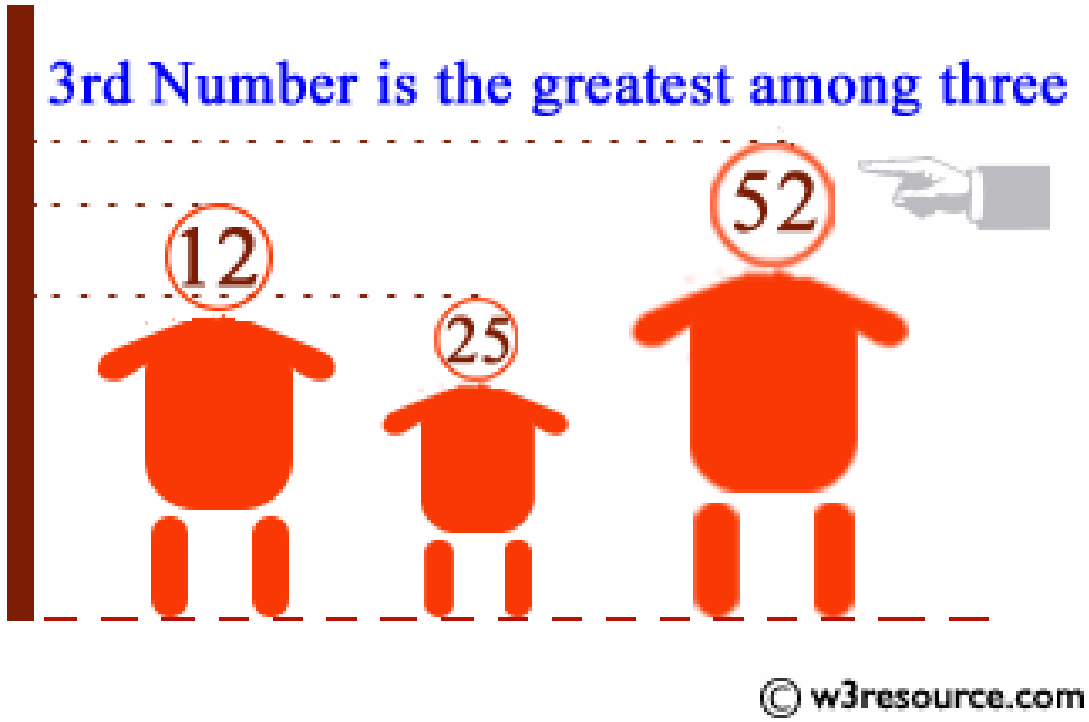
- Draw flow chart and Write a C program to accept the height of a person in centimeters and categorize the person according to their height.
- Categorization criteria is given below:

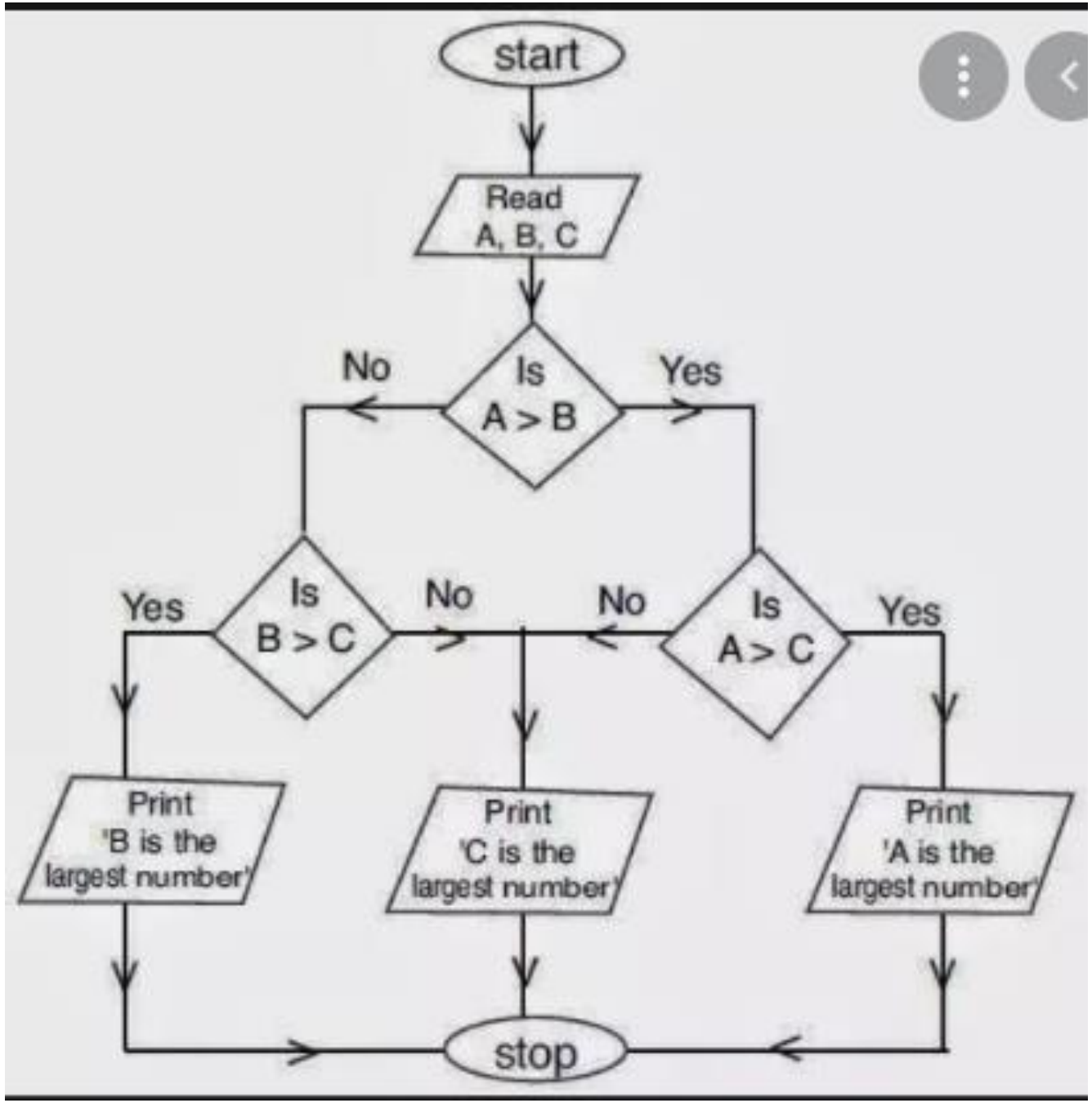



```
#include <stdio.h>
void main()
{
   float PerHeight;
    printf("Input the height of the person (in centimetres) :");
   scanf("%f", &PerHeight);
   if (PerHeight < 150.0)
      printf("The person is Dwarf. \n");
   else if ((PerHeight >= 150.0) && (PerHeight < 165.0))
      printf("The person is  average heighted. \n");
   else if ((PerHeight >= 165.0) && (PerHeight <= 195.0))
      printf("The person is taller. \n");
   else
      printf("Abnormal height.\n");
}
```

Question

- Write a C program to find the largest of three numbers.




start
Read
A, B, C
Is
A > B
No
Yes
Is
B > C
Yes
No
No
Is
A > C
Yes
Print
'B is the
largest number'
Print
'C is the
largest number'
Print
'A is the
largest number'
stop

Answer

```
#include <stdio.h>
void main()
{
   int num1, num2, num3;

   printf("Input the values of three numbers : ");
   scanf("%d %d %d", &num1, &num2, &num3);
   printf("1st Number = %d,\t2nd Number = %d,\t3rd Number = %d\n", num1, num2, num3);
   if (num1 > num2)
   {
      if (num1 > num3)
      {
printf("The 1st Number is the greatest among three. \n");
      }
      else
      {
         printf("The 3rd Number is the greatest among three. \n");
      }
   }
   else if (num2 > num3)
      printf("The 2nd Number is the greatest among three \n");
   else
      printf("The 3rd Number is the greatest among three \n");
}
```

Alternative Algorithm

Control Statements Cond….

- Exercise: Draw a flow chart, Write pseudo code to determine the grades
- Hint : marks >=90 “A GRADE”

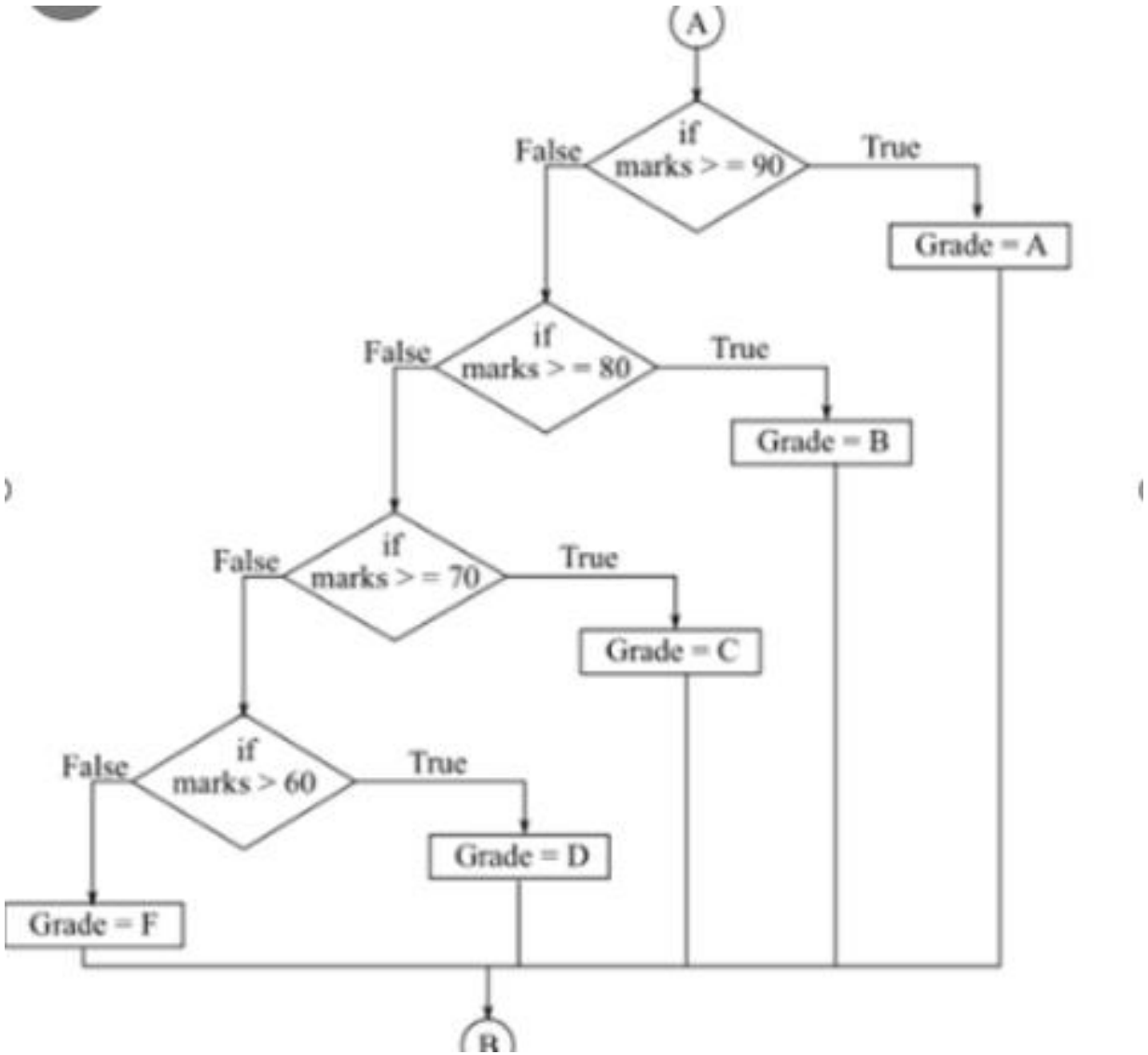
  marks >=80 and =< 89 “B GRADE”

  marks >=7 0and =< 79 “C GRADE”

  marks >=60 and =< 69 “D” GRADE”

  Otherwise “ F GRADE”

Flow Chart


A
if
marks > = 90
False
True
Grade = A
if
marks > = 80
False
True
Grade = B
if
marks > = 70
False
True
Grade = C
if
marks > 60
False
True
Grade = D
Grade = F
B

Pseudocode Algorithm

```
Read marks
if marks >=90 then
        print( "A Grade")
Else if  (marks >=80)  then
        print (" B Grade")
Else if  (marks>= 70 ) then
         print ("C Grade")
Else if  marks >=60 then
         print (“D Grade")
else
         print ("F Grade ")
endif
```

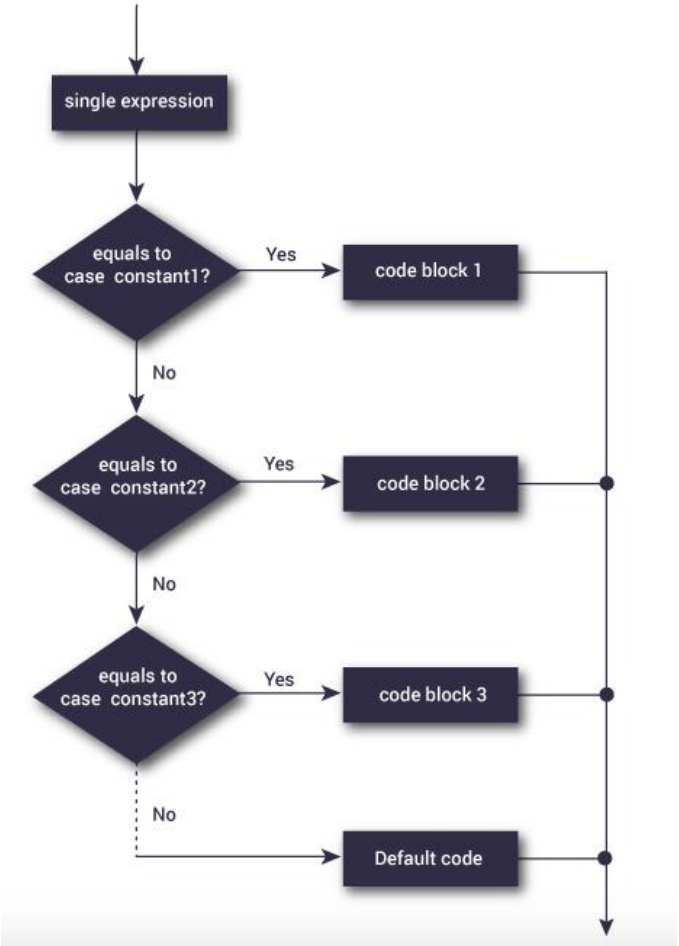
Example: Grades

- Write a program to give the grade when you enter your mark of a subject.
- The grade of the mark is defined as follows.
  - 00 – 24: E
  - 25 – 34: D
  - 35 – 49: C
  - 50 – 69: B
  - 70 – 100: A

Example: Grades

```c
int main() {

    int mark;
    char grade;

    printf("Enter your mark: ");
    scanf("%d", &mark);

    if(mark>69) {
        grade = 'A';
    }
    else if(mark>49) {
            grade = 'B';
    }
    else if(mark>34) {
            grade = 'C';
    }
    else if(mark>24) {
        grade = 'D';
    }
    else {
        grade = 'E';
    }
    printf("your grade is %c", grade);
}
```

case Statement



- The if..else..if ladder allows you to execute a block code among many alternatives.
- If you are checking on the value of a single variable in if...else...if, it is better to use switch statement.
- The switch statement is often faster than multiple if...else.

Example:

```c
int main () {

   char grade;

   printf("Enter your grade: ");
   scanf("%c", &grade);

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }

   printf("Your grade is  %c\n", grade);

   return 0;
}
```

Rules of case statement

1. The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
2. You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
3. The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
4. When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.
5. When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
6. Not every case needs to contain a **break**. If no **break** appears, the flow of control will fall through to subsequent cases until a break is reached.
7. A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.

Example: Grades

```c
int main() {

    int mark;
    char grade;

    printf("Enter your mark: ");
    scanf("%d", &mark);

    switch (mark) {
        case 0 ... 24:
            grade = 'E';
            break;
        case 25 ... 34:
            grade = 'D';
            break;
        case 35 ... 49:
            grade = 'C';
            break;
        case 50 ... 69:
            grade = 'B';
            break;
        case 70 ... 100:
            grade = 'A';
            break;
        default:
            grade = 'F';
    }
    printf("your grade is %c", grade);
}
```

Loops

- Loops are used in programming to repeat a specific block of code.
- There are three loops in C programming:
  1. for loop
  2. while loop
  3. do...while loop

For loop

- It also executes the code until condition is false.
- In this three parameters are given that is:
- Initialization
- Condition
- Increment/Decrement

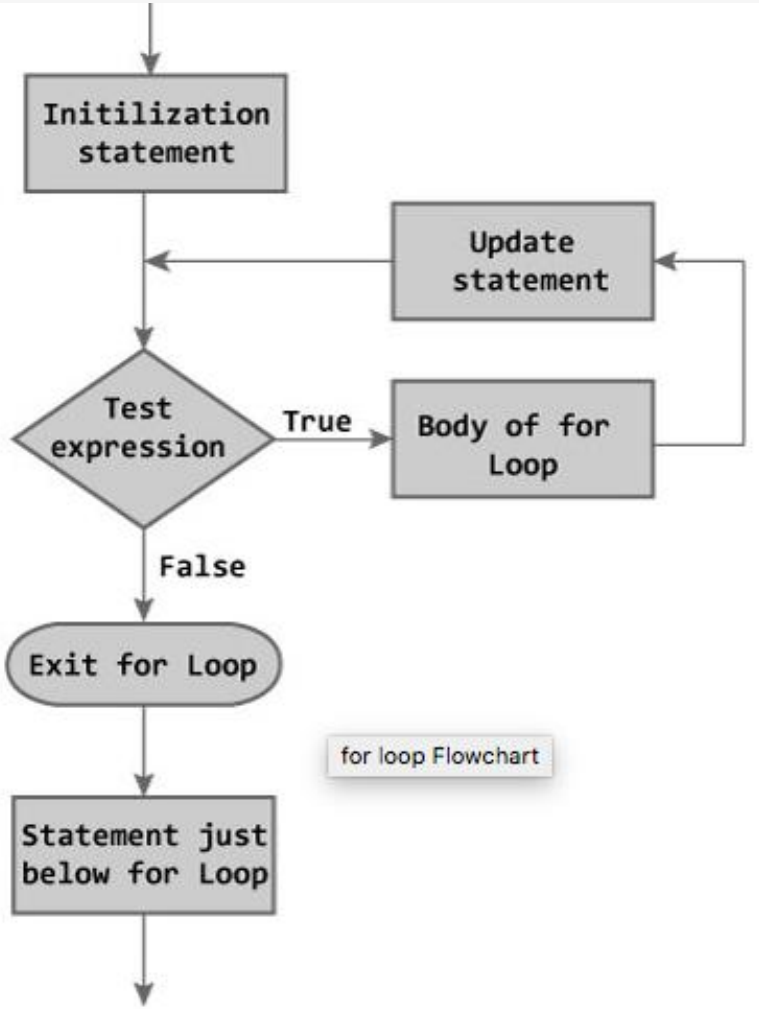
Syntax:

```
for (initialization; condition; increment/decrement) {

    // Code statements to be executed

}
```

for Loop

```
for (initializationStatement; testExpression; updateStatement)
{
        // codes
}
```

1. The initialization statement is executed only once.
2. Then, the test expression is evaluated. If the test expression is false (0), the loop is terminated.
3. If the test expression is true (nonzero), codes inside the body of the loop is executed.
4. Then, the update statement is executed and update the variable.
5. This process repeats **until the test expression is false**.
6. The for loop is commonly used **when the number of iterations is known**.



for loop Flowchart

Example

```
#include<stdio.h>

void main()

{

int i;

for( i = 20; i < 25; i++) {

printf ("%d " , i);

}

}
```

Output:

```
20 21 22 23 24
```

Example: Write a C program to Find Factors

- Example :

- If number is 20 ,
- then factors of 20 are :1,2,4,5,10,20

Write a C program to Find Factors

```
#include <stdio.h>
int main() {
    int num, i;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    printf("Factors of %d are: ", num);
      for (i = 1; i <= num; ++i) {
        if (num % i == 0) {
            printf("%d ", i);
        }
       }
       return 0;
}
```

Nested for Loops

- It is also possible to place a loop inside another loop. This is called a **nested loop**.
- The "inner loop" will be executed one time for each iteration of the "outer loop":

Write down the output of the following program

```
#include <stdio.h>
int main() {
  int i, j;
    // Outer loop
  for (i = 1; i <= 2; ++i) {
    printf("Outer: %d\n", i);  // Executes 2 times

    // Inner loop
    for (j = 1; j <= 3; ++j) {
      printf(" Inner: %d\n", j);  // Executes 6 times (2 * 3)
    }
  }
    return 0;
}
```

Write down the output of the following program

```
#include <stdio.h>
int main() {
  int i, j;
    // Outer loop
  for (i = 1; i <= 2; ++i) {
    printf("Outer: %d\n", i);  // Executes 2 times


    // Inner loop
    for (j = 1; j <= 3; ++j) {
      printf(" Inner: %d\n", j);  // Executes 6 times (2 * 3)
    }
  }
    return 0;
}
```

```
Outer: 1
 Inner: 1
 Inner: 2
 Inner: 3
Outer: 2
 Inner: 1
 Inner: 2
 Inner: 3
```

Example: Finding Factors

```c
int main() {

    int number, i;

    printf("Enter a positive number: ");
    scanf("%d", & number);
    printf("Factors of %d are:\n", number);

    for (i = 1; i < number/2; ++i) {
        if(number%i==0) {
            printf("%d x %d = %d\n", i, number/i, number);
       }
    }

    printf("done!");

   return 0;
}
```