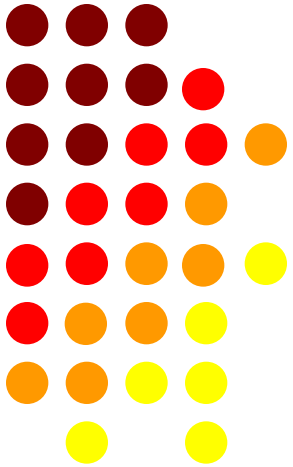
Programming For Problem Solving

Lecture 10



Operators & Its Classification

- **Operators** are special symbols whose meaning is already known to C compiler. There are 45 operators in C classified as:
- **Unary Operator:** Operators that need only one operating value or operand to complete its task is termed as unary operator. Example: (**!**) logical not (**~**) complement.
- **Binary Operators:** Operators that need two operand to complete its task is termed as binary operator. Example + (Addition), * (Multiplication)
- **Ternary Operators:** Operators that need three operand to performed it task is termed as conditional operator. Example exp1?exp2:exp3
- It first evaluate the exp1 condition, if it is true then exp2 is evaluated, if the condition is false then exp3 is evaluated

Operators & Its Classification (Cont..)

- **The operators are classified in eight general categories**
- Arithmetic Operator
- Relational Operator
- Logical Operator
- Assignment Operator
- Increment / Decrement Operator
- Bitwise Operator
- Conditional Operator
- Special Operator

Operators & Its Classification (Cont..)

- **Arithmetic Operators:** This operators which help us to carryout basic arithmetic operations are termed as arithmetic operators such addition, subtraction, multiplication, division

Operator	Meaning	Examples
+	Addition	1+2 = 3
-	Subtraction	3-2 = 1
*	Multiplication	2*2 = 4
/	Division	2/2 = 1
%	Modulo division	10%3 = 1

Operation	Result	Examples
Int/Int	Int	5/2 = 2
Real/Int	Real	5.0/2 = 2.5
Int/Int	Int	5%2 = 1
Real/Int	Real	5.0%2 = Error
Int/Int	Int	-5%-2 = -1

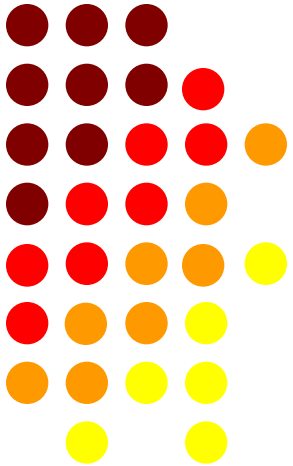
Operators & Its Classification (Cont..)

- **Relational Operators:** The operators which are used to form conditions for comparing two operands or values are termed as relational operator. There are six relational operators used in C

Operator	Meaning	Example	Return value
<	is less than	3<5	1
<=	is less than or equal to	4<=2	0
>	is greater than	7>5	1
>=	is greater than or equal to	3>=5	0
==	equal to	6==6	1
!=	not equal to	5!=5	0

Programming For Problem Solving

Lecture 11



Operators & Its Classification (Cont..)

- **Logical Operators:** The operators which are used to combine the results of two or more conditions are termed as logical operator. There are 3 logical operators used in C

Operator	Meaning	Example	Return value
&&	Logical And	(9>2) && (6>4)	1
||	Logical OR	(9>2) || (3>4)	1
!	Logical Not	!4	0

AND (&&)		
T	T	T
T	F	F
F	T	F
F	F	F

OR (||)		
T	T	T
T	F	T
F	T	T
F	F	F

Operators & Its Classification (Cont..)

- **Assignment Operators:** The operators which are used to assign the right hand side computed value to left hand side variable is termed as assignment operator.
- **Syntax:** identifier = expression; like int r=2, ac; ac=3.14*r*r
- **Increment/Decrement Operator:** The operator which is used to increment or decrement the value of variable by one is termed as increment/decrement operator. Example ++, --

Pre-Inc/Dec Operator	Post-Inc/Dec Operator
Operator comes before the operand	Operator comes after the operand
Value is incremented first & then it is assigned	Value is assign first & then it is incremented
int x=2, y; y=++x Then: x=3, y=3	int x=2, y; y=x++ Then: x=3, y=2

Operators & Its Classification (Cont..)

- **Bitwise Operators:** The operators which are used to perform operation at bit level are termed as bitwise operator. There are six bitwise operators used in C

Operator	Meaning	Example	Return value
&	Bitwise AND	5&7	5
|	Bitwise OR	5|7	7
^	Bitwise XOR	5^7	2
~	Complement	~5	-6
<<	Left Shift	4<<2	16
>>	Right Shift	16>>1	8

Bitwise operator contd…

1. **Bitwise AND**
   - 1 & 1 = 1
   - 1 & 0 = 0
   - 0 & 1 = 0
   - 0 & 0 = 0
2. **Bitwise OR**
   - 1 | 1 = 1
   - 1 | 0 = 1
   - 0 | 1 = 1
   - 0 | 0 = 0
3. **Bitwise XOR**
   - 1 ^ 1 = 0
   - 1 ^ 0 = 1
   - 0 ^ 1 = 1
   - 0 ^ 0 = 0

Eg: x = 3 = 0000 0011
y = 4 = 0000 0100
x&y = 0000 0000

Eg: x = 3 = 0000 0011
y = 4 = 0000 0100
x|y = 0000 0111

Eg: x = 3 = 0000 0011
y = 4 = 0000 0100
x ^ y = 0000 0111

Bitwise Left shift Operator

- The Left shift operator (<<) shifts each bit of the operand to its Left. The general form or the syntax of Left shift operator is
- variable << no. of bits positions
- if x = 7 (i.e., 0 0 0 0 0 1 1 1) the value of y in the expression
- y = x <<1 is 14
- 0 0 0 0 1 1 1 0 = 14 since it shifts the bit position to its left by one bit. The value stored in x is multiplied by 2^N (where n is the no of bit positions) to get the required value. For example, if x = 7 the result of the expression y = x << 2 is y = x * 2^2 (i.e. 28)

Bitwise Right shift Operator

- The Right shift operator (>>) shifts each bit of the operand to its Right. The general form or the syntax of Right shift operator is

  variable >> no. of bits positions

  if x = 7 (i.e., 0 0 0 0 0 1 1 1) the value of y in the expression

- y = x >> 1 is 3

- 0 0 0 0 0 0 1 1 = 3 since it shifts the bit position to its right by one bit. The value stored in x is divided by 2^N (where n is the no of bit positions) to get the required value. For example, if x = 7 the result of the expression y = x << 2 is y = x / 2^2 (i.e. 1). If you use the left shift operator i.e. x = x << 1 the value of x will be equal to 2 (i.e., 0 0 0 0 0 0 1 0) since the lost bit cannot be taken back.

Bitwise 1's Complement & 2' Complement

- The one's complement operator (~) is a unary operator, which causes the bits of the operand to be inverted (i.e., one's becomes zero's and zero's become one's)
- For Example, if x = 7

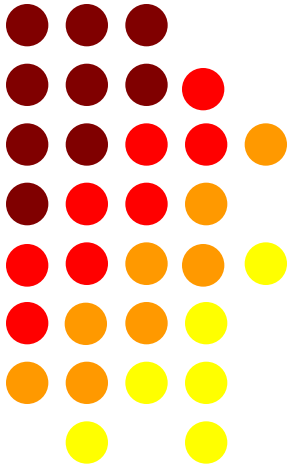
  i.e. 8 – bit binary digit is 0 0 0 0 0 1 1 1
- The One's Complement is 1 1 1 1 1 0 0 0

Operators & Its Classification (Cont..)

- **Special Operators:** The operators like comma, sizeof are termed as special operator.
- **Comma Operator:** Comma operator is used to separate multiple values in an expression or a statement
- Like int i=2, j;
- j = i + (1,2,3,4,5);
- j=7
- **Sizeof Operator:** sizeof operator is used to find the number of bytes occupied by a datatype, variable or a value.
- Like int i;
- sizeof(int)=2 sizeof(i)=2 sizeof(5)=2

Programming For Problem Solving

GROUP OF INSTITUTIONS

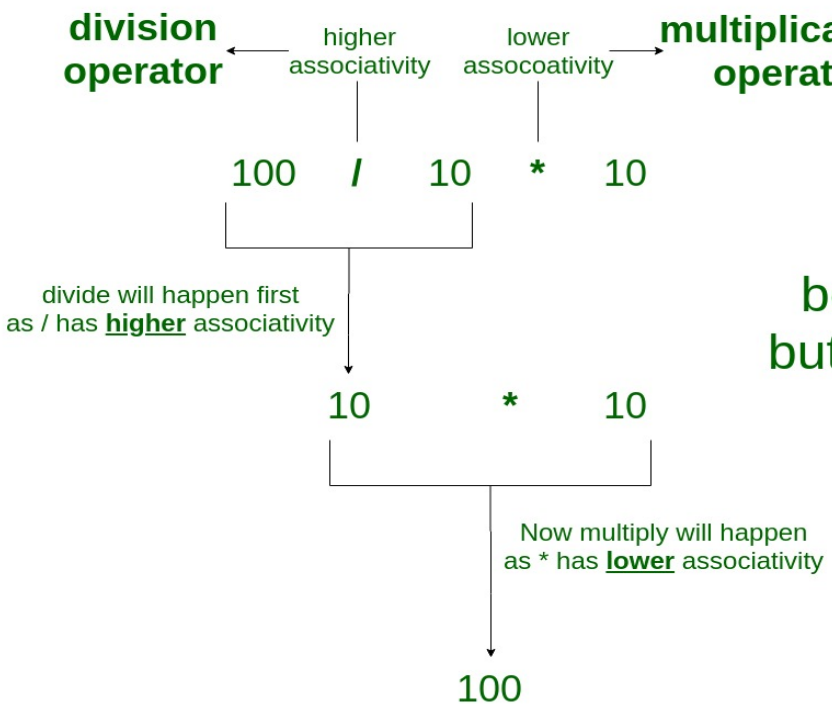


Lecture 12

Precedence & Associativity of Operator

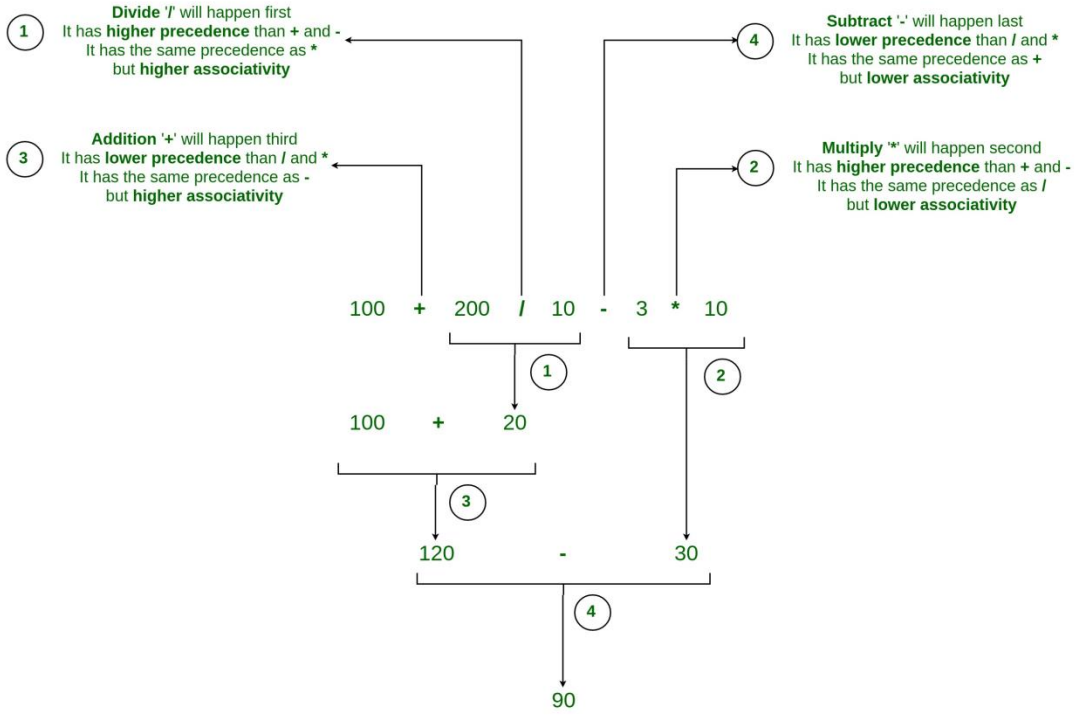
- **Precedence** is a term which describes the order of execution of operators in an expression having different priority. The highest precedence operator is applied first, followed by the next highest, and so on.
- For example * has high precedence than +.
- **Associativity** is a term which describes the order of execution of operators in an expression having same priority. It tell that how the operator of same precedence are grouped and how the expression will be evaluated.
- For example arithmetic operator are left associative but assignment operator are right associative.

Operator Associativity



/ and ***
both have the same precedence
but Left to Right (**LTR**) associativity

Operator Precedence and Associativity



/ and ***** both have the same precedence but Left to Right (**LTR**) associativity

+ and **-** both have the same precedence but Left to Right (**LTR**) associativity

/ and ***** have the higher precedence than **+** and **-**

Precedence & Associativity of Operator

Operator	Meaning of operator	Associativity	Priority
() [] -> .	Functional call Array element reference Indirect member selection Direct member selection	Left to right	1
! ~ + - ++ -- & * sizeof (type)	Logical negation Bitwise(1 's) complement Unary plus Unary minus Increment Decrement Operator(Address) Pointer reference Returns the size Type cast(conversion)	Right to left	2

Precedence & Associativity of Operator

Operator	Meaning of operator	Associativity	Priority
* / %	Multiply Divide Remainder	Left to right	3
+ -	Binary plus(Addition) Binary minus(subtraction)	Left to right	4
<< >>	Left shift Right shift	Left to right	5
< <= > >=	Less than Less than or equal Greater than Greater than or equal	Left to right	6
== !=	Equal to Not equal to	Left to right	7

Precedence & Associativity of Operator

Operator	Meaning of operator	Associativity	Priority
&	Bitwise AND	Left to right	8
^	Bitwise exclusive OR	Left to right	9
|	Bitwise OR	Left to right	10
&&	Logical AND	Left to right	11
||	Logical OR	Left to right	12
?:	Conditional Operator	Right to left	13
=,*=,/=,%=,-=,&=,^=,|=,<<= >>=	Assignment Operator	Right to left	14
,	Comma operator	Left to right	15

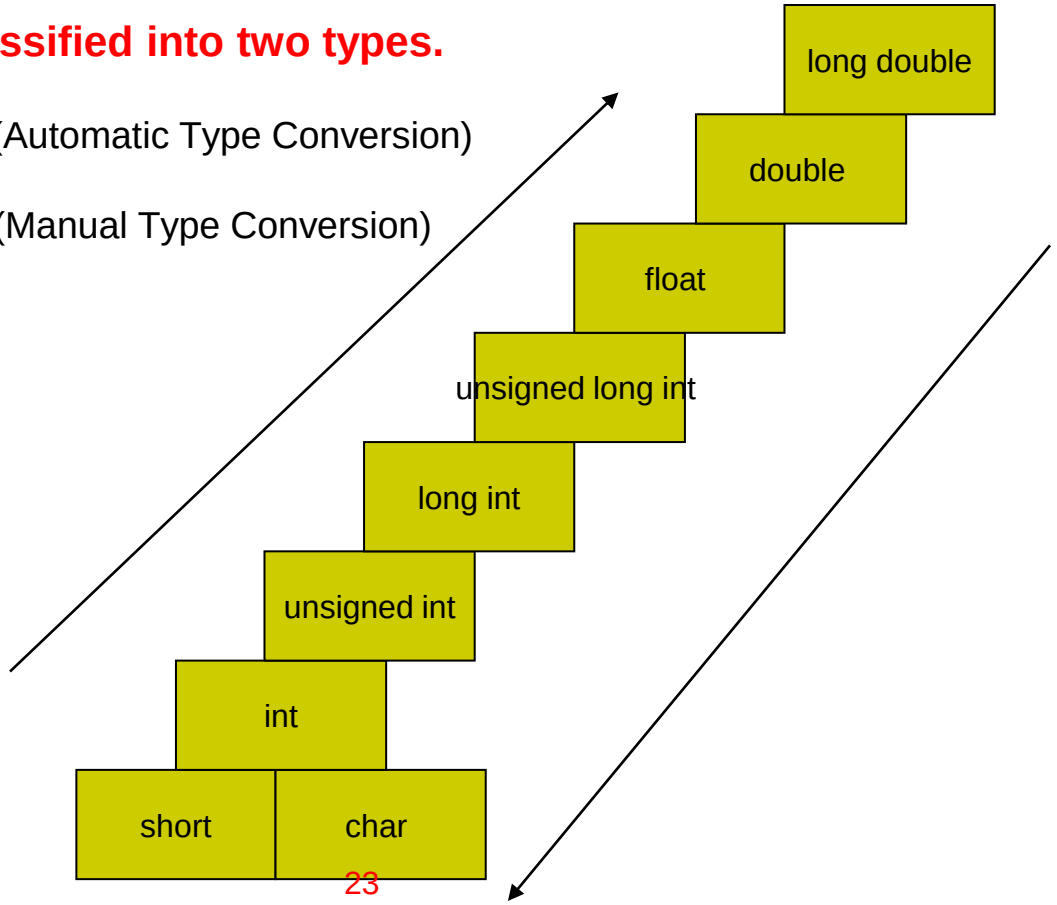
Precedence & Associativity of Operator

- **Example**
- Y= 4 * 2 / 4 – 6 / 2 + 3 % 2 * 6 / 2 + 2 > 2 && 4 ! = 2
- = 8 / 4 – 6 / 2 + 3 % 2 * 6 / 2 + 2 > 2 && 4 ! = 2
- = 2 – 6 / 2 + 3 % 2 * 6 / 2 + 2 > 2 && 4 ! = 2
- = 2 – 3 + 3 % 2 * 6 / 2 + 2 > 2 && 4 ! = 2
- = 2 – 3 + 1 * 6 / 2 + 2 > 2 && 4 ! = 2
- = 2 – 3 + 6 / 2 + 2 > 2 && 4 ! = 2
- = 2 – 3 + 3 + 2 > 2 && 4 ! = 2
- = -1 + 3 + 2 > 2 && 4 ! = 2
- = 2 + 2 > 2 && 4 ! = 2
- = 4 > 2 && 4 ! = 2
- = 1&& 4 ! = 2
- = 1 && 1 = 1

Role of Type Conversion in C

- Type Casting means One data type converted into another data type. This is called Type conversion or Type casting.
- **Type conversion is classified into two types.**

1. Implicit Type Conversion (Automatic Type Conversion)
2. Explicit Type Conversion (Manual Type Conversion)



Implicit conversion

- The Implicit Type Conversion is known as Automatic Type Conversion.
- C automatically converts any intermediate values to the proper type so that the expression can be evaluated without losing any significance.
- Implicit type Conversion also known as Converted Lower order data type into Higher order data type.
- Implicit Type Conversion also known as **Widening.**

For Example:

int a, b;	float a,b;
float c;	int c;
c = a + b;	c=a+b; // wrong assignment
Print c;	Print c;

Explicit conversion

- The Explicit Type Conversion is, there are instances when we want to force a type conversion in a way that is different from the automatic conversion. The Explicit Type Conversion is Converted Higher order data type into Lower order data type.
- The Explicit type Conversion is also known as borrowing.
- The Explicit type conversion forces by a casting operator.

syntax

(type_name) expression;

Where type_name is one of the standard C data type.The expression may be a constant, variables or an expression.

For Example:

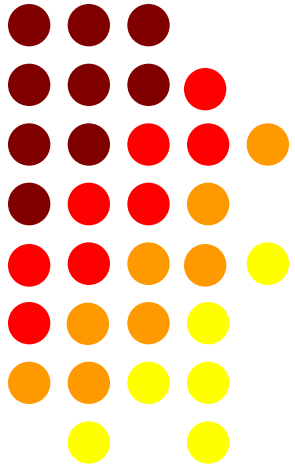
- float a, b;
- int c;
- c = (int) a + (int) b;
- Print c;

Role of Type Conversion in C

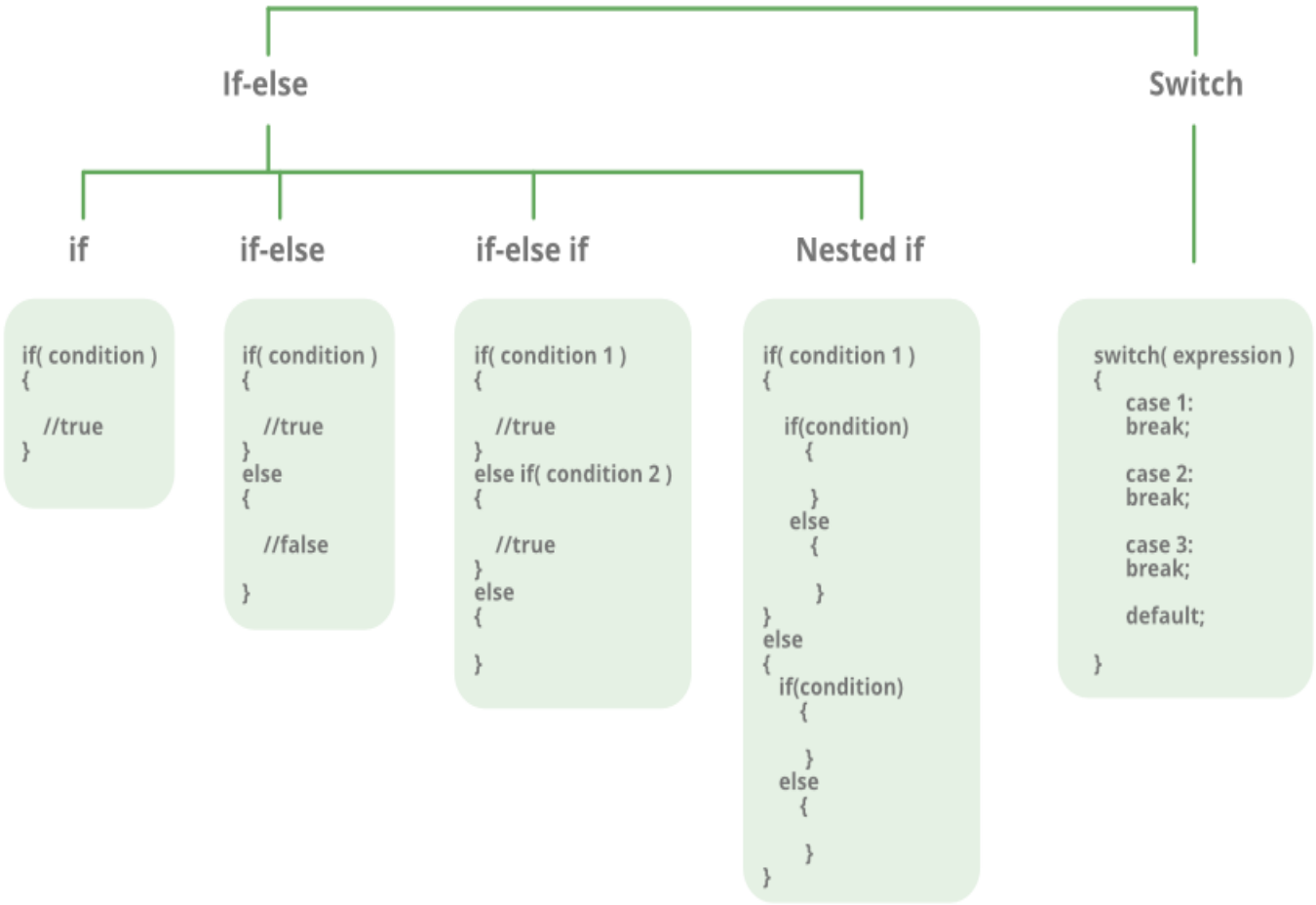
Implicit Type Conversion	Explicit Type Conversion
It is a Automatic Type Conversion	It is a Manual Type Conversion
It is performed in lower to higher datatype only	It can be performed in any order
float i; i=5.0 / 2 = 2.5 Here 5.0 belongs to double datatype 2 belongs to int datatype So 2 get converted in to double before execution	float i; i=(int)5.0 / 2 = 2.0 Here 5.0 belongs to double datatype 2 belongs to int datatype But 5.0 get converted in to int before execution

Programming For Problem Solving

Lecture 13



Decision Making



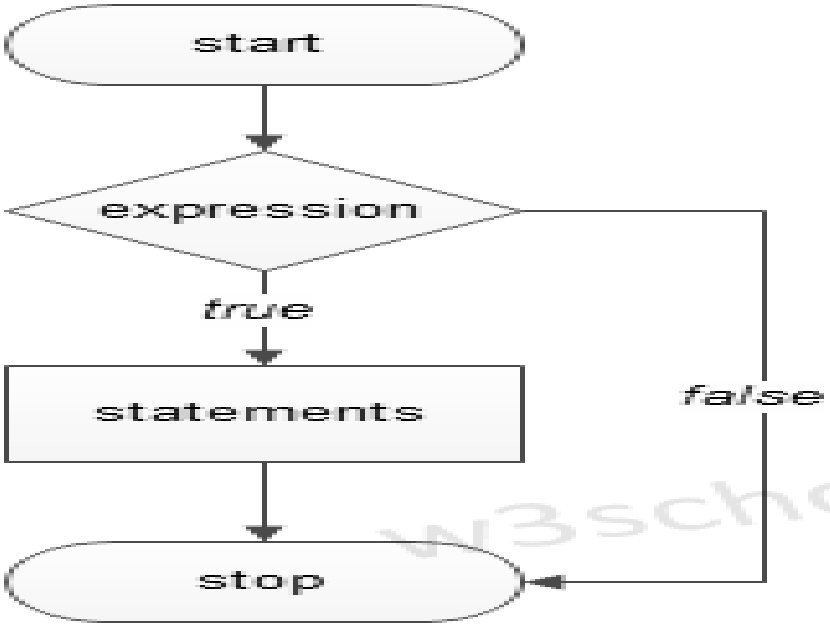
Simple if

- If the expression evaluates to **true**, then the block of code inside the 'if' statement will be executed.
- If the expression evaluates to **false**, then the first set of code after the end of the 'if' statement (after the closing curly brace) will be executed.
- C programming language assumes any **non-zero** and **non-null** values as **true** and if it is either **zero** or **null**, then it is assumed as **false** value.

syntax

```
if(expression)
    {
    statements ;
    }
```

IF STATEMENTS



Simple if example

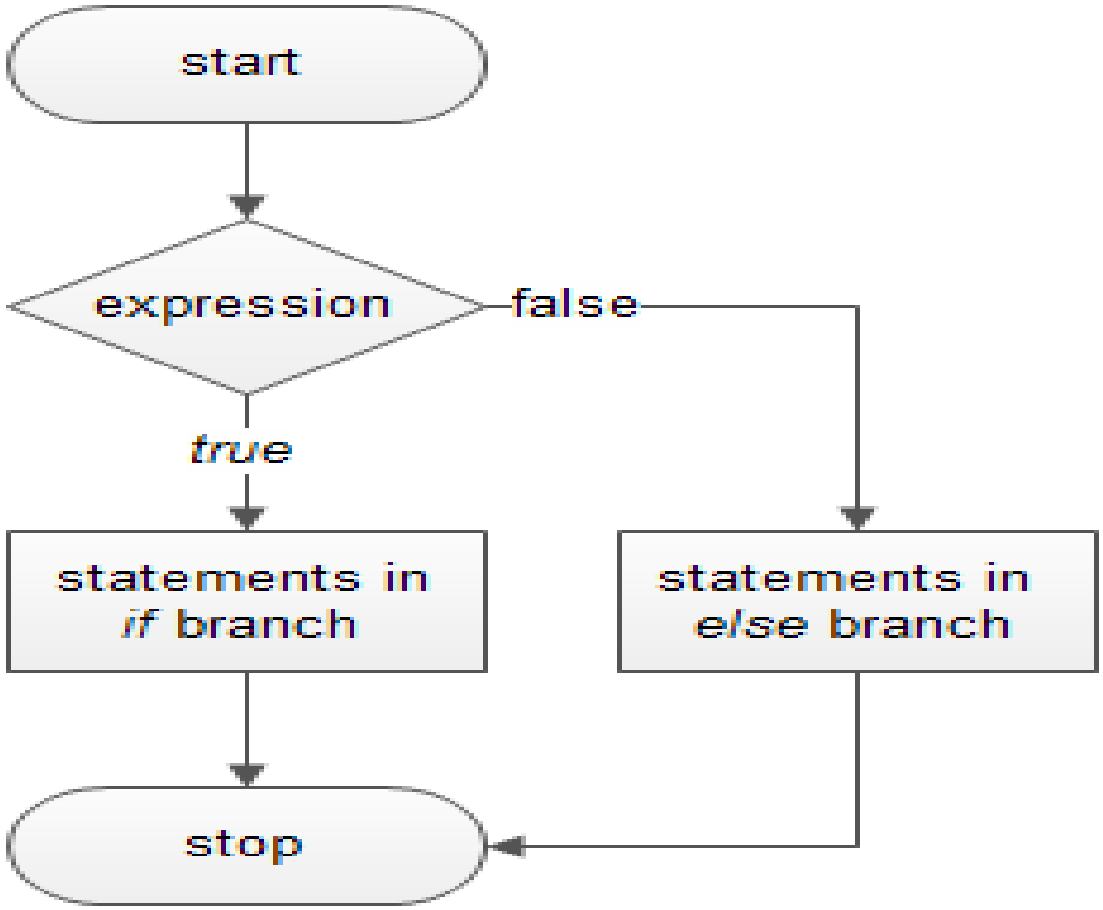
```
#include<stdio.h>
int main()
{
int quant,cost;
Printf(”\n Enter number of items and cost per item \n”);
Scanf(“%d%d%”,&quant,&cost);
bill=quant*cost;
if(bill>=3000)
{
bill=bill-500;
printf(“You will get 500RS Discount”);
}
return 0;
}
```


If else statement

- The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't.
- But what if we want to do something else if the condition is false. Here comes the C else statement.
- else is optional statement.
- We can use the else statement with if statement to execute a block of code when the condition is false.

General Syntax

```
if(expression)
{
   statement 1;
}
else
 {
     statement 2;
}
```

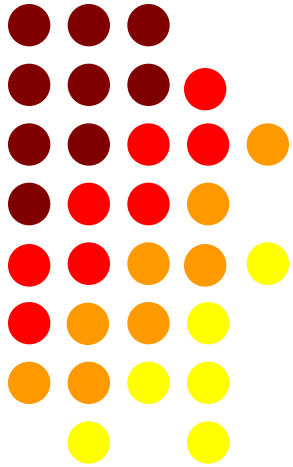

start
expression
false
true
statements in
if branch
statements in
else branch
stop

Example:- WAP to print even or odd

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int n;
clrscr();
printf(“enter the no”);
scanf(“%d”,&n);
if(n%2==0)
printf(“\n%dis even no”);
else
printf(“\n%d is odd no”);
getch();
}
```


Programming For Problem Solving

Lecture 14

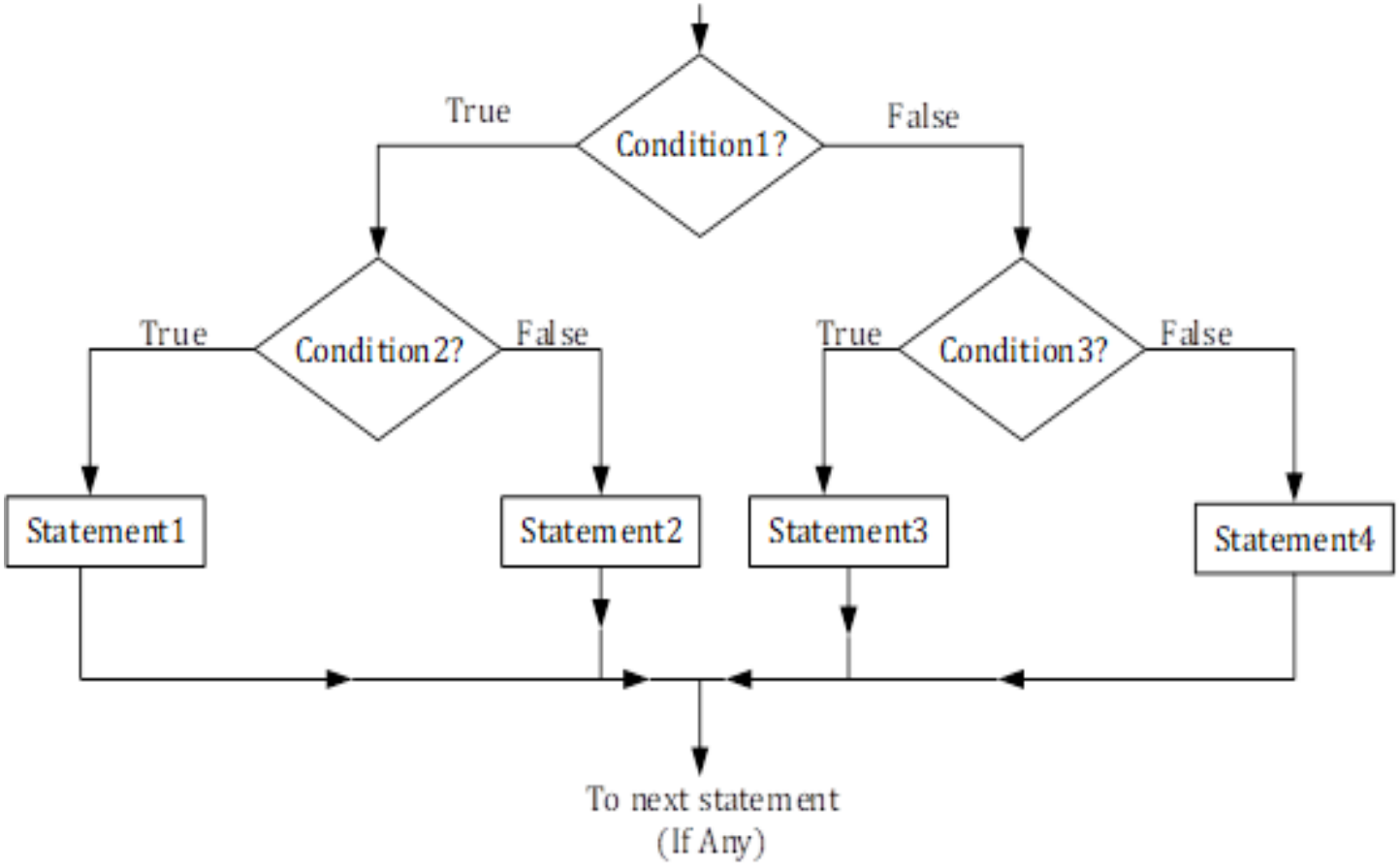


Nested if else

- Which means you can use if or else statement inside another if or else block.
- Code needs to be executed to match the corresponding if and else and pair of braces.
- if the condition is true it goes to inner if , and statements will execute, other wise statement in else block will execute.

Gernal Syntax

```
if(condition1)
{
        if (condition2)
                        {
            Statement 1;
            }
        else
            {
            Statement 2;
            }
}
else
        {
            if(condition 3)
            {
                statement 3;
            }
            else
            {
                statement 4;
            }
}
```


Condition1?
True
False
Condition2?
True
False
Condition3?
True
False
Statement1
Statement2
Statement3
Statement4
To next statement
(If Any)

Program to find largest of three number

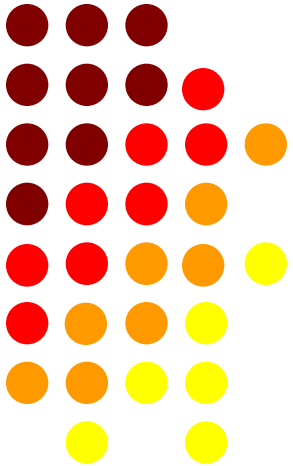
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf"\n enter \n");
Scanf("%d%d%d",&a,&b,&c);
if(a>b)
{
if(a>c)
   {
printf("\n%dislargest",a);
  }
else{
printf("\n %d is largest",c);
}}
else
{
  if(b>c)
  {
printf("\n %d is largest",b);
 }
else
{
printf{"\n%d is largest,c);
}}
getch();
}
```

WAP to check year is leap or not

```
#include<stdio.h>
int main()
{
int yr;
printf(“\n enter year\n”);
scanf(“%d”,&yr);
If(yr%100==0)
{
  if(yr%400==0)
  {
   printf(“\n %d is leap yr”,yr);
  }
  else
   {
printf(“\n%d is not a leap yr”,yr);
}}
else
{
        if(yr%4==0)
              {
             printf(“\n %d is leap yr”,yr)
              }
        else
              {
              printf(“\n %d is not leap yr”,yr);
}
}
return 0;
```

Programming For Problem Solving

Lecture 15



Else if ladder

-In C programming language the else if ladder is a way of putting multiple ifs together when multipath decisions are involved.

-It is a one of the types of decision making and branching statements.

-A multipath decision is a chain of if's in which the statement associated with each else is an if.

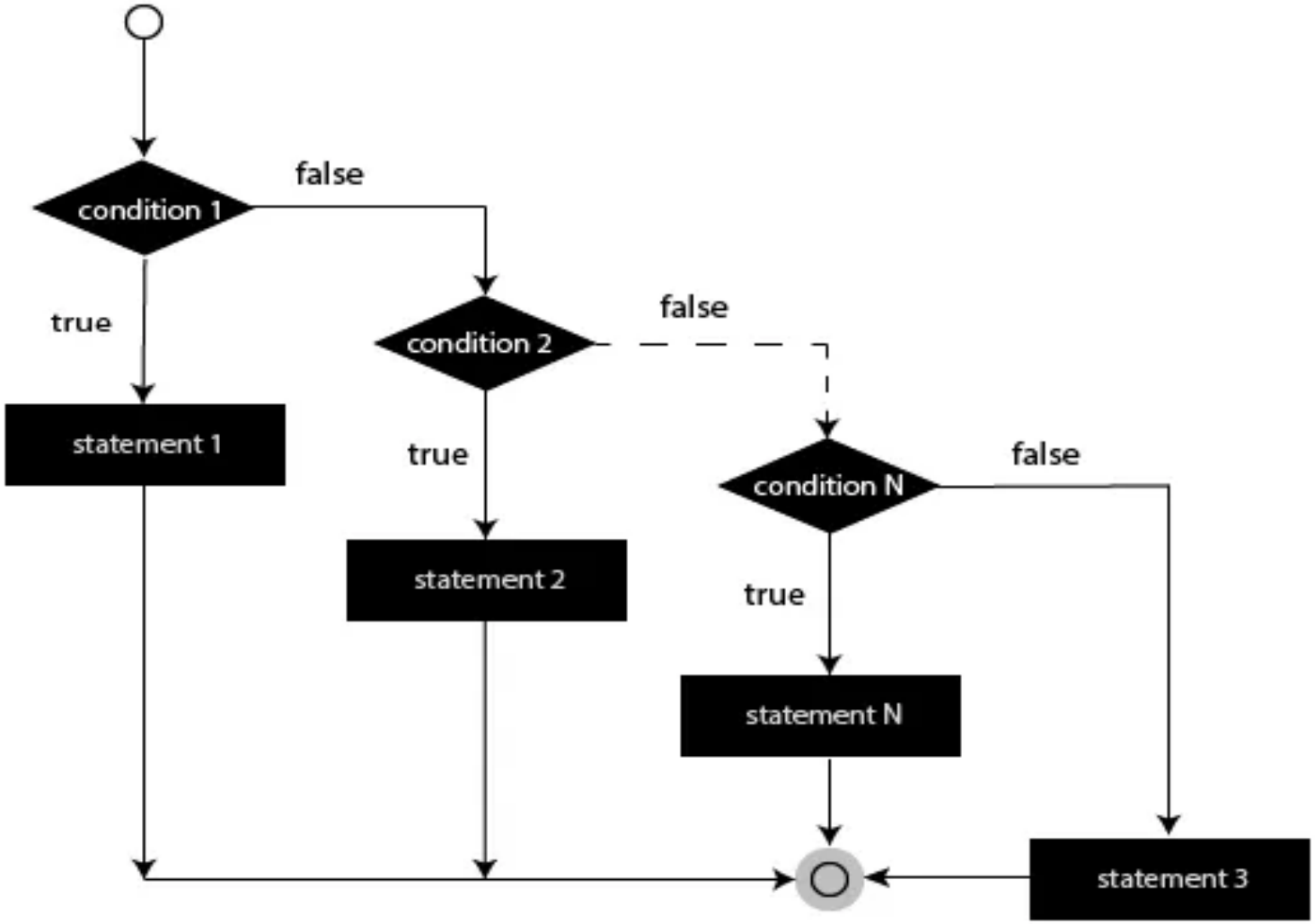
-The if – else – if statement is also known as the if-else-if ladder or the if-else-if staircase.

-The conditions are evaluated from the top to downwards

Else if ladder

Syntax:-

```
if(condition1)
statement1;
else if(condition2)
statement 2;
else if (condition3)
statement3;
      -
       -
      -
else
    statement n;
```

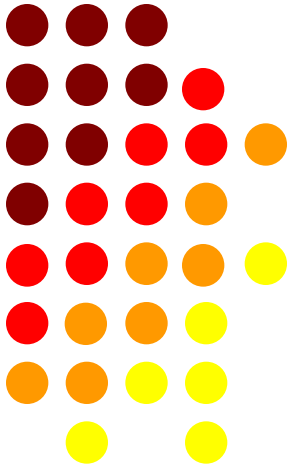


condition 1
false
true
statement 1
condition 2
false
true
statement 2
condition N
false
true
statement N
statement 3

Example

```
#include <stdio.h>
    int main()
    {
        int x;
        x = 0;
        clrscr ();
        printf("Enter Choice (1 - 3) : ");
        scanf("%d", &x);
        if (x == 1)
            printf ("\nChoice is 1");
        else if ( x == 2)
            printf ("\nChoice is 2");
        else if ( x == 3)
            printf ("\nChoice is 3");
        else
            printf ("\nInvalid Choice ");
     return 0;
    }
```


Programming For Problem Solving

Lecture 16



SWITCH STATEMENT

It is a in built multiway decision system in C.

The control statement that allows us to make a decision from the number of choices is called the switch case statement.

Rules for switch statement

- the switch case must be constant or a constant expression.
- the case label must be constant and unique.
- case label must end with colon(:) and each statement with semi colon(;)
- case label can be int or char constant but it cannot be float.
- using break and default is optional.

Syntax

Syntax:-

```
switch(integer exp)
{
case value1:
        block 1;
            break;
case value2:
          block 2;
          break;
case value n:
           block n;
           break;
default:
          block x;
}
```


Use of break and default with switch.

- You can use the break statement to end processing of a particular labeled statement within the switch statement.
- It branches to the end of the switch statement. Without break, the program continues to the next labeled statement, executing the statements until a break or the end of the statement is reached.
- This continuation may be desirable in some situations.

Use of default with switch

- The default statement is executed if no case constant-expression value is equal to the value of expression.
- If there's no default statement, and no case match is found, none of the statements in the switch body get executed.
- There can be at most one default statement.
- The default statement doesn't have to come at the end. It may appear anywhere in the body of the switch statement.
- A case or default label can only appear inside a switch statement.

PROGRAM TO DESIGN A CALCULATOR

```
#include <stdio.h>
int main()
{
int a,b,c,ch;
printf("\nEnter 1 for addition:\n ");
printf("Enter 2 for subtraction:\n ");
printf("Enter 3 for multiply:\n");
printf("Enter 4 for division:\n ");
scanf("%d",&ch);
printf("Enter a number:\n");
scanf("%d",&a);
printf("Enter second number:\n");
scanf("%d",&b);
switch(ch)
{
case 1 : c=a+b;
        printf(“sum is :%d\n",c);
         break;
case 2 : c=a-b;
        printf("Sub is : %d\n",c);
         break;
case 3 : c=a*b;
          printf(“Mul is%d\n",c);
           break;
case 4 : c=a/b;
      printf("div is : %d\n",result);
     break;
default: printf("wrong input\n");
}
    return 0;
 }
```

Find Output

```
#include <stdio.h>
#include<conio.h>
 void main()
 {
 int num=2;
switch(num+2)
 {
 case 1:
printf("Case1: Value is: %d", num);
case 2:
 printf("Case1: Value is: %d", num);
case 3:
 printf("Case1: Value is: %d", num);
default:
printf("Default: Value is: %d", num);
 }
getch();
 }
```

Calculator

```
#include <stdio.h>
#include<conio.h>
    void main()
    {
    char operator;
    int num1,num2;
    printf("\n Enter the operator (+, -, *, /):");
    scanf("%c",&operator);
    printf("\n Enter the Two numbers:");
    scanf("%d%d",&num1,&num2);
    switch (operator)
    {
    case '+':
    printf("%d+%d=%d",num1,num2,num1+nu
    m2);
    break;

    case '-':
    printf("%d-%d=%d",num1,num2,num1-
    num2);
    break;

case '*':
    printf("%d*%d=%d",num1,num2,num1*
    num2);
    break;

    case '/':
    printf("%d / %d =
    %d",num1,num2,num1/num2);
    break;

    default:
    printf("\n Enter the operator only");
    break;
}
getch();
    }
```


THANK YOU :)