DELTA Pg No.
Date / /

value	int i;	float f;
—	gr	gr
5/2	2	2.0
5.0/2	2	2.5
5/2.0	2	2.5
5.0/2.0	2	2.5
2/5	0	0.0
2.0/5	0	0.4

Control Flow Statements

→ The execution flow of the prog. is under control of control flow statements.

→ In C prog. lang. control-flow statements are classified into 3 types

1) Selection statements
   ex :- else, if, elseif, switch
2) Iteractive statements
   ex - while, for, do while
3) Jumping statements
   Ex- break, continue, goto

(1) Selection Statements

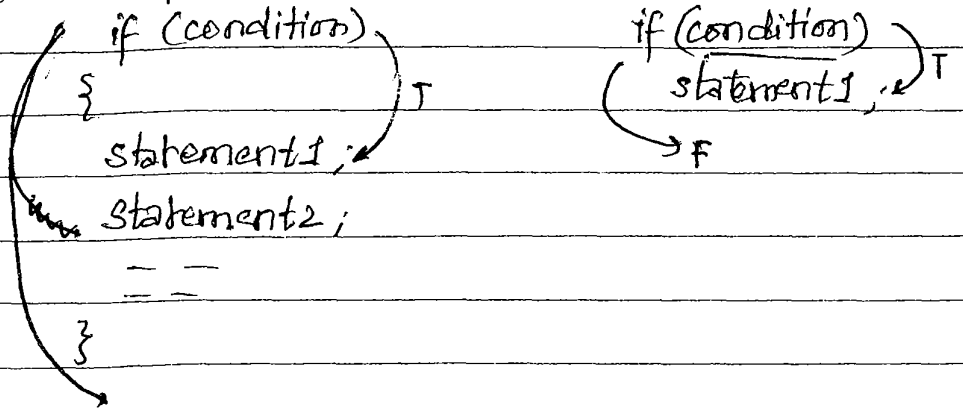
→ These are also called decision making statements

→ By using them, we can create conditional oriented block.

→ When we are working with selection statements if condition is true then block is executed if condition is false then corresponding block will be ignored

DELTA Pg No.
Date / /

Syntax to If:-



→ Constructing the body is always optional.

→ Body is recommended to use when we having multiple statements

→ For a single statement, it doesn't require to specify of body, if body is not mentioned then automatically scope is terminated with next semicolon.

* Else :-

→ else is a keyword, by using this keyword we can create alternate block of if condition

→ Using else is always optional, it is recommended to use when we having alternate block.

→ When we are working with if and else, only one block can be executed i.e when 'if' condition is false then only 'else' part is executed

Syntax

```
if (condition)
{
statem
```

DELTA Pg No.
Date / /

Syntax

(1) if (condition) → T
{
statement 1
F statement 2
}
else
{
statement 3
statement 4
}

In this
In 'if' - multiple statements
'else' - multiple statements

(2) if (condition) → T
{
statement 1
F statement 2
}
else
statement 3

if → ms
else → s

(3) if (condition) → T
statement 1;
else
F statement 2;

if - s
else s

(4) if (condition) → T
statement 1;
F else
{
Statement 2;
statement 3;
}

if - s
else - ms

PROGRAM

void main()
{
printf("A");
printf("B");
if (2 < 5) // if (1 && 1)
{
printf("C");
printf("D");
}

WARNING MESSAGE

When

```
    printf ("Welcome");
}
```

O/P : ABNITC Welcome

Note: When we are constructing any conditions by using constant expression then compiler provides a WARNING MESSAGE i.e condition is always true or False

→ When the warnings are occured, it can be ignored if it is possible bcz prog. can be executed properly.

PROGRAM

(1)

```
void main()
{
  printf ("NIT");
if ( 1 != 2<5 || 0 == 5<8)        // if (1 != 1 || 0 == 1)
                                     0 || 0 = 0
  {
    printf ("A");
F   printf ("B");
  }
 printf ("C");
}
```

O/P : NITC

(2)
```
Void main ()
{
 printf ("Hello");
 if (5<8 != 1)        // (1 != 1) ⇒ (0 = 1) False
   printf ("A");  → if scope is close here
 printf ("B");
 printf ("C");
}
```
O/P : HelloBC

→ When the body is not specified then automatically scope is terminated with next semicolon i.e within the condition only 1 statement will be placed

(3)
```
Void main()
{
 printf ("NIT");
 if (2<5 != 2>5)      // (1 != 0) True
{
 printf ("A");
 printf ("B");
}
 else
  {
   printf("C");
   printf ("D");
}
}
```
O/P: NITAB

DELTA Pg No.
Date / /

(4)
```
void main()
{
 printf ("Welcome");
  if (!5)                    // 0 - False
  {
    printf ("A");
    printf ("B");
  }
else
  {
    printf ("C");
    print ("D");
  }
}
```

O/P: Welcome CD

5)
```
void main()
 {
   printf("A");
   if (5>8!=1).          // if (0!=1)
   {
     printf ("B");
     printf ("C");
   }
else
  {
    printf ("D");  -> else scope is close here.
    printf ("NIT");
}
```

O/P: ABCNIT

→ When the body is not specified for else part then automatically scope is terminated with next semicolon.

6)
```
Void main
{
  printf("A")
  if (
     printf("B");
     printf("C");
  else
     {
       printf("NIT");
       printf("C");
     }
}
```

O/P: ~~ABC~~ Error, misplaced else

else scope starts only after if scope only.

→ According to syntax of if-else when we are using elsepart it must be required to start after if scope only.

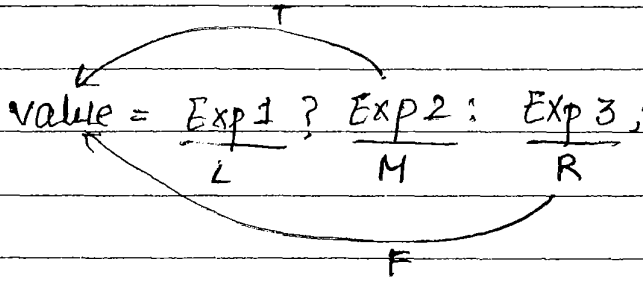
CONDITIONAL OPERATORS (?:) →

1. Conditional Operators are ternary category operators.
2. Ternary Category means it required 3 arguments, i.e left, middle & rightside arguments.
3. When we are working with conditional Operators, if cond^n is true, then returns with middle arg. If cond^n is false, then returns with rightside arg.

DELTA Pg No.
Date / /

and left side argument is treated like condition.

Syntax:



4. According to syntax, if exp 1 is true or leftside value is non-zero, then exp2 or middle value is ~~written~~ returned.

5. If exp1 is false or leftside value is 0, then exp3 or right side value is returned.

6. When we are working with conditional operators, we require to satisfy following conditions —
   1) No. of ? and : should be equal.
   2) Every colon should match with just before ?.
   3) Every " " followed by ? only.

• int a;

1. a = 10 ? 20 : 30; O/P = 20

   a = 10 (L) ? 20 (M) : 30 (R) ;

2. a = 5 < 8 (1) ? 20 : 30; O/P = 30

3. a = 2 > 5 ? 10 : 20 : 30; O/P = Error

   No. of ? and : are equal

DELTA | Pg No. | Date / /

4. `a = 2<5!=1 ? 10 : 5>2 ? 20 : 30;`

O/P : L 1 m 1 — 2 2 — R

upto 1st ? → L

after 1st ? Before 1st : → m

after 1st : Before ; → R

`1 != 1 | 5>2 ? 20 : 30`

L m R

O/P = 20

* 1. `a = 5>2 ? 2<5! = 0 ? 10 : 20 : 30;`

L M R

`a = 5>2 ? 2<5! = 0 ? 10 : 20 : 30;`

1 — 2 2 1

L m R

O/P → 10

* inta

1. `a = 5>8 ? 10 : 2<5! = 1 ? 20 : 5>2 ? 30 : 40;`

L 1 m 1 — R 2 2 3 3

L m R

L m R

O/P → 30

`2<5! = 1 ? 20 : 5>2 ? 30 : 40`

L m R

False → then right side

`5>2 ? 30 : 40`

L m R

True then m (30)

DELTA Pg No.
Date / /

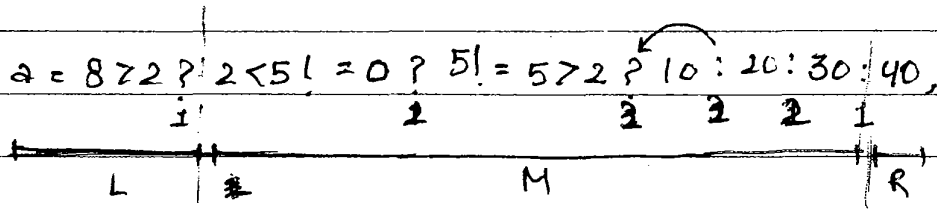
→ When we are working with multiple ? and : then initially we required to convert the expression into 3 arguments.

→ In order to convert the expression into 3 arguments, it is recommended to follow Numbering System

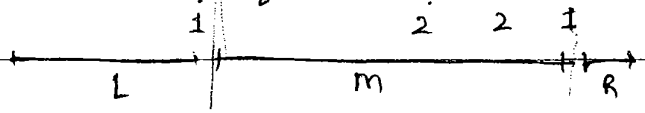
→ According to numbering process, for every ? one unique no. required to assign and for every ':' corresponding ? no. required to assigned.

→ Upto 1st question mark, it is called Left argument after 1st ?, b4 1st ? corresponding : should be middle argument and remaining complete part is Right side argument

1 a = 8>2 ? 2<5!=0 ? 5! = 5>2 ? 10 : 20 : 30 : 40;



2<5!=0 ? 5! = 5>2 ? 10 : 20 : 30



5! = 5>2 ? 10 : 20

L m R

5! = 1

O/P : 10

2. a = 5 > 8 ? 2 < 5 ? 10 : 20 : 2 > 5 ! = 0 ? 30 : 5 < 8 ? 2 > 5 ! = 0 ? ! 5 ? 40 : 8 ? 50 : 60 : 70 : 80;

1 2 2 1 3 3 4 5 6 6 7 7 5 4

L m R

L m R

5 < 8 ? 2 > 5 ! = 0 ? ! 5 ? 40 : 8 ? 50 : 60 : 70 : 80

L M R

2 > 5 ! = 0 ? ! 5 ? 40 : 8 ? 50 : 60 : 70

L M R

O/P : 70

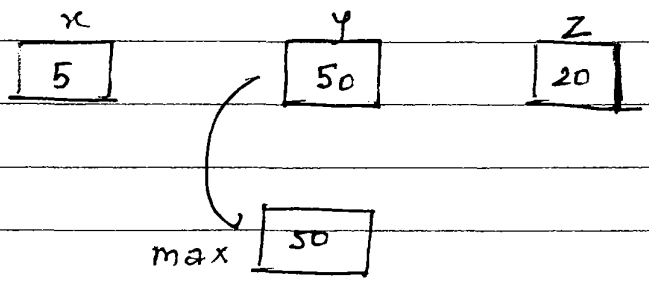
DELTA Pg No. Date / /

DELTA Pg No.
Date / /

1) The basic advantage of conditional operator is reducing coding part of the prog.

2) When we are reducing the coding part then it occupies less memory, so automatically performance will increase.

PROGRAM




```
void main()
{
  int x, y, z, max;
   x = 5; y = 50; z = 20;
 if (x > y && x > z)
  {
    max = x;
  }
 if (y > x && y > z)
  {
    max = y;
  }
 if (z > x && z > y)
  {
   max = z;
  }
 printf ("max value is: %d", max);
}
```

C/P : Max value is: 50

- In implementation when interelated blocks are constructed independently then after completion of the requirement also, compiler checks remaining all cond's. So it is time taking process

- When interrelated blocks are occur then always recommended to create in optimal blocks by using else part.

- By using else part, we can create only 1 optimal block, if we required to create multiple blocks then recommended to go for nested if else

Note: In previous prog., in place of writing multiple condns we can create single statement which can provide max value i.e conditional operator required to use.

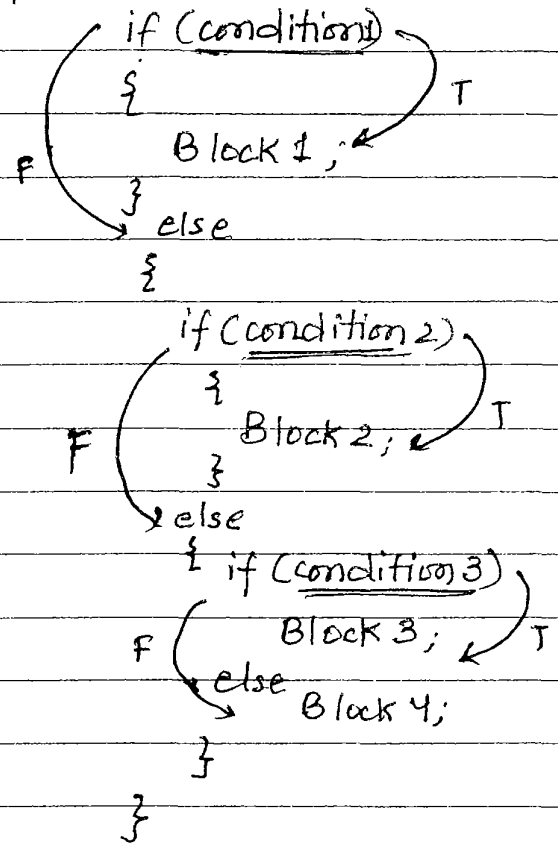
*
```
void main()
{
  int x, y, z, max;
  x=5; y=50; z=20;

  max = x>y&&x>z ? x : y>z ? y : z;

  printf("Max value is %d", max);
}
```

DELTA | Pg No.
Date / /

NESTED If-else

* It is a procedure of constructing a condn within an existing conditional block.
* In C prog. lang., it is possible to place upto 255 nested blocks.

Syntax:

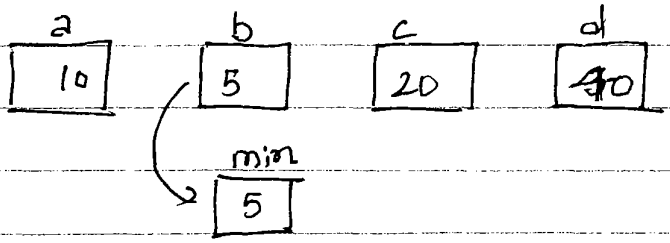


* According to syntax, if condition 1 is true then block 1 is executed, if it is false then control will pass to else part.
* Within the else part if condition 2 is true then block 2 will executed, if it is false then control will pass to nested else.
* Within the nested else, if condition 3 is true then block 3 will executed, if it is false then block 4 will be executed.

DELTA | Pg No. | Date / /

* When we are working with nested if else at any point of time, only one block will be executed or can be executed.
* Nested concepts can be applied for if part and else part also
* If we are applying the nested concepts to if part then it is called nested if-else, if we are applying to else part, then it is called else if ladder

— Prog:




```
void main()
{
 int a,b,c, d , min ;
 clrscr();
 Printf (" Enter 4 values : ");
 Scanf (" %d %d %d %d ", &a, &b, &c, &d);
 if (a<b && a<c && a<d)
  {
    min = a;
   }
 else
   {
     if (b<c && b<d)
        { min = b;
        }
```

```
else
  {
    if (c<d)
      min = c;
    else
      min = d;
  }
}
Printf ("min value is : %d", min);
getch();
}
```

O/P: Enter 4 values : 10 20 30 40
min value is : 10

* Scanf

- It is a predefined func^n which is declared in stdio.h
- By using scanf function we can read d data from user.
- When we are working with scanf function, it can take any no. of arguments, but first argument must be string const & remaining arguments r seperated with ',' .
- When we are working with scanf() func^n within the double quotes, we are required to pass proper format specifier only i.e. what type of data we are reading, same type of format specifier is required.
- Scanf() func^n will works with the help of call by address mechanism thats why every variable requires '&' symbol.