Unit -2

C Programming

DATE : / 20
PAGE No

* Designed by Dennis Ritchie at Bell Laboratories

* first Phase:
- Writing C instruction
- saving file with .c extension

* second phase:
- Compile it into object code or machine language code ~~fde~~ using C Compiler.
- Object file with .obj extension created

* final phase:
- Building the executable Program, called linking
- file with .exe Extension created

* Data type:

Data type	Detail
Char	1 byte with range -128 to 127
int	2 byte with range -32768 to 32767

DATE: / 20
PAGE No

unsigned	2 byte with range 0 to 65535
long int	4 byte with range -2147483648 to 2147483647
float	4 bytes with range 10-88 to 1038 with 7 digit of Precision.
double	8 bytes with range 10-308 to 10308 with 15 digits of precision
void	generic pointer, used to indicate no function parameters

DATE : / 20
PAGE No

* Variables :-

A named piece of memory which is used to hold a value which may be modified by the program.

* Attributes of variable :
- types
- value
- address

* Naming variable :
- must be a valid identifier
- must not be a keyword
- Names are case sensitive
- variables are identified by only first 32 characters

* Declaring variable :
- Declaration announces that data type of a variable and allocates appropriate memory location.
- Each variable used must be declared.
- data_type vaible_list ;

DATE: / 20
PAGE No

* Initializing variable;
- Declaring variable initialized it with garbage value by default.

- data_type vaible_name = Constt;

* Constant:
fixed value that cannot be altered by the Program and can be number, characters or strings

Const type	Example
char	'a' '$', '1'
int	10, 0, -15
unsigned int	0, 250
long int	32767657
float	12.23456
double	1433.345345454
String	"Hello word"

DATE : / 20
PAGE No

* Escape sequency :

Have spacial meaning.

Type	Meaning
\n	New line
\t	Tab
\b	Backspace
\'	Print Single Quote
\"	Print double Quote
\0	Null character

* Input / output :

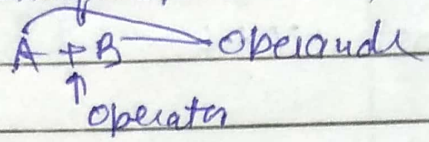
I/O specifiers

Specifier	Detail
%.c	character
%.d or %.i	integer or signed int.
%.u	unsigned integer
%.l or %.ld	Long or signed Long
%.lu	unsigned long
%.f	floating point
%.lf	Double floating point
%.s	String

DATE: / 20
PAGE No

Expression:

Combination of operator & operand.



operators:

(i) Arithmetic operators:

Arithmetic operators can further be classified as unary operators and binary operators.

Symbol	Stand for	example
+	addition	x + y
-	Substraction, negation	x - y, -x
*	multiplication	x * y
/	Division	x/y
%	modulus or remainder	x % y
--	Decrement	--x, x--
++	Increment	++x, x++

DATE : / 20
PAGE No

Increment -

(i) Pre ++x

(ii) Post x++

Decrement

(i) Pre --x

(ii) Post x--

* Relational and logical Operators.

A relational operator is used to compare two values and the result of such an operation is always logical, i.e either true or false

The valid relational operators supported by C are given below

Symbol	Stands for	Example
>	Greater than	x>y
>=	Greater than equal to	x>=y
<	less than	x<y
<=	less than equal to	x<=y
==	Equal to	x==y
!=	Not equal to	x!=y

* Logical operators:

And &&

OR ||

NoT !

X	!X
T	f
f	T

X	Y	X && Y	X || Y
T	T	T	T
T	f	f	T
f	T	f	T
f	f	F	f

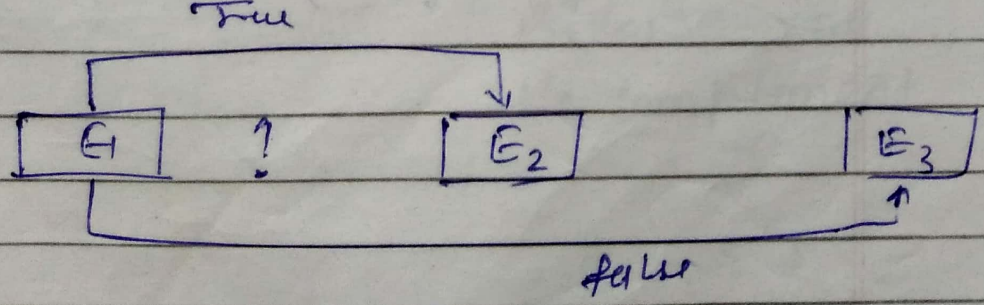
* Conditional operators:

- The General form of this operation is

  E1 ? E2 : E3;

  where E1, E2 and E3 are expressions.

- Also called Ternary operators



DATE : / 20
PAGE No

* Assignment operator :
Assign the value of the expression on the right hand side to a variable on the left hand side of the assignment operator (=).

- <variable name> = <expression>

- multiple Assignment
a = b = c = 30;

- Compound Assignment
A += B;
A = A + B;

* Size of operator :
- Returns the no. of byte the operand (variable, Const^n or data type) occupies in the main memory.

- int a = 10;
int size;
size = size of(a);

DATE: / 20
PAGE No

* Comma Operator :

- used to string together a no. of expressions which are performed in a sequence from left to right

- Ex :

```
a = (x=5, x+2);
```

executes in the following order :

(i) a value 5 is assigned to variable x.
(ii) x is incremented by 2.
(iii) the value of expression x+2 (i.e 7) is assigned to the variable a.

* Bitwise operators :

- used for the manipulation of data items at bit level.

Operator	stands for
<<	Left shift
>>	Right shift
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	1's complement

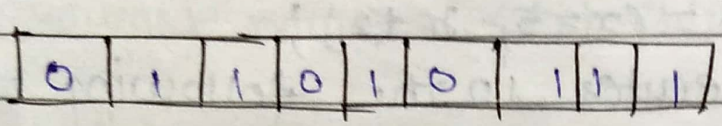
DATE : / 20
PAGE No

- Left shift (<<):

No. of bits to be shifted left

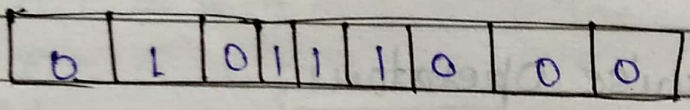
```
x<<3;
```

x before Left shift



Lost bits

x after Left shift

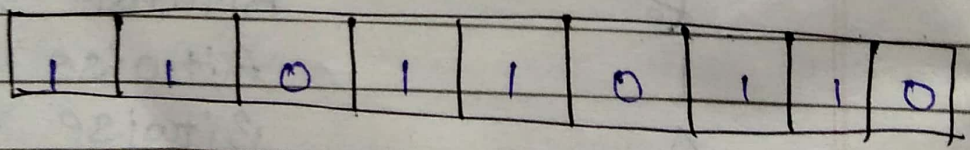


appended bits

- Right Shift : (>>):

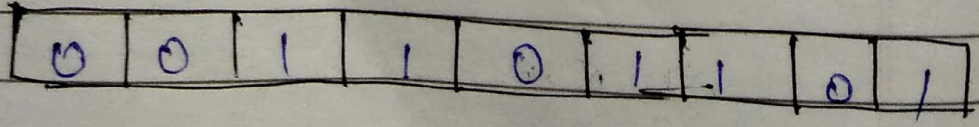
```
y>>2
```

y before right shift



Lost bits

y after right shift



appended bits

DATE: / 20
PAGE No

- Bitwise AND (&) : 178 & 63 = 50

```
  1 0 1 1 0 0 1 0   (178)
& 0 0 1 1 1 1 1 1   (63)
= 0 0 1 1 0 0 1 0   (50)
```

- Bitwise OR (|) :

```
  1 0 1 1 0 0 1 0   (178)
| 0 0 1 1 1 1 1 1   (63)
= 1 0 1 1 1 1 1 1   (191)
```

* Bitwise XOR (^) :

```
  1 0 1 1 0 0 1 0   (178)
^ 0 0 1 1 1 1 1 1   (63)
= 1 0 0 0 1 1 0 1   (141)
```

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Both same = 0
Both diff = 1.

DATE : / 20
PAGE No

* 1's Complement (~) :

	1	0	1	1	0	0	1	0 (178)
~178 =	0	1	0	0	1	1	0	1 (77)

* 2's Complement = 1's Complement + 1

* Arithmetic with characters :

- Use ASCII codes

- char ch;
  ch = 'A';
  ch = ch + 1;
  printf("ch = %c", ch);

- The output of this program segment would be
  ch = B

DATE: / 20
PAGE No

* Type conversion / casting :
• Implict Type Conversion :
- Automatic type conversion
- Done by the compiler on its own
- data types of the variable are upgraded to the data type of the variable with largest data type
- implict conversions may lose information.

• Explicit Type Conversion :
- User defined
- Syntax : (data-type) expression

* variables types :
• Local Variable :
- Declared inside a function or block
- Can be used only by statements that are inside that function or block of code.

- Not known to function Outside their Own.

• Global variable :
- defined Outside a function, usually on top of the program
- hold their values throughout the life time of Program
- Can be accessed by any function.

DATE : / 20
PAGE No

* Storage classes :
- Decides the part of storage to allocate memory for a variable
- Also determines the scope of a variable

Storage classes in C

Storage specifier	Storage	Initial value	Scope	Life
auto	stack	Garbage	within block	End of block
extern	Data segment	zero	Global multiple file	Till End of Prg
static	Data segment	zero	within block	"
register	CPU register	Garbage	"	End of block

* Auto:
- Default storage class for all the variables declared inside a function or a block

* Register:
- used to define local variables that should be storage in a register instead of Ram

DATE: / 20
PAGE No

:- **Static:**
Preserve the value of their last use in their scope

* **extern:**
used to give a reference of a global variable that is visible to all the program files.

*

DATE : / 20
PAGE No

* Conditional / decision statements:

Helps in making decision based on certain conditions

- Types:
  - If statement
  - if else statement
  - Nested if else statement
  - if else if ladder

If statement:

- Executes the code if condition / expression is true
- Also called one way selection statement
- Syntax:

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

DATE: / 20
PAGE No

(2) If-else statement :

- Executes 'if' statement if Condition is true otherwise 'else' statement.
- Also called two way selection statement
- Syntax :

```
if (condition)
{
   statement1;
   ...
}
 else
{
  statement2;
  ...
}
```

(3) Nested if-else statement :

- Include if-else statements inside the body of another if-else statement.

- used when Program requires more than one test expression.

- Also called multi way selection statement.

DATE : / 20
PAGE No

- Syntax.

```
if (condition 1)
{
    if (condition 2)
    {
        Statement-block-1
    }
  else
    {
      statement block 2;
    }
}
  else
  {
     Statement block 3.
  }
```

(4) if-else if ladder:

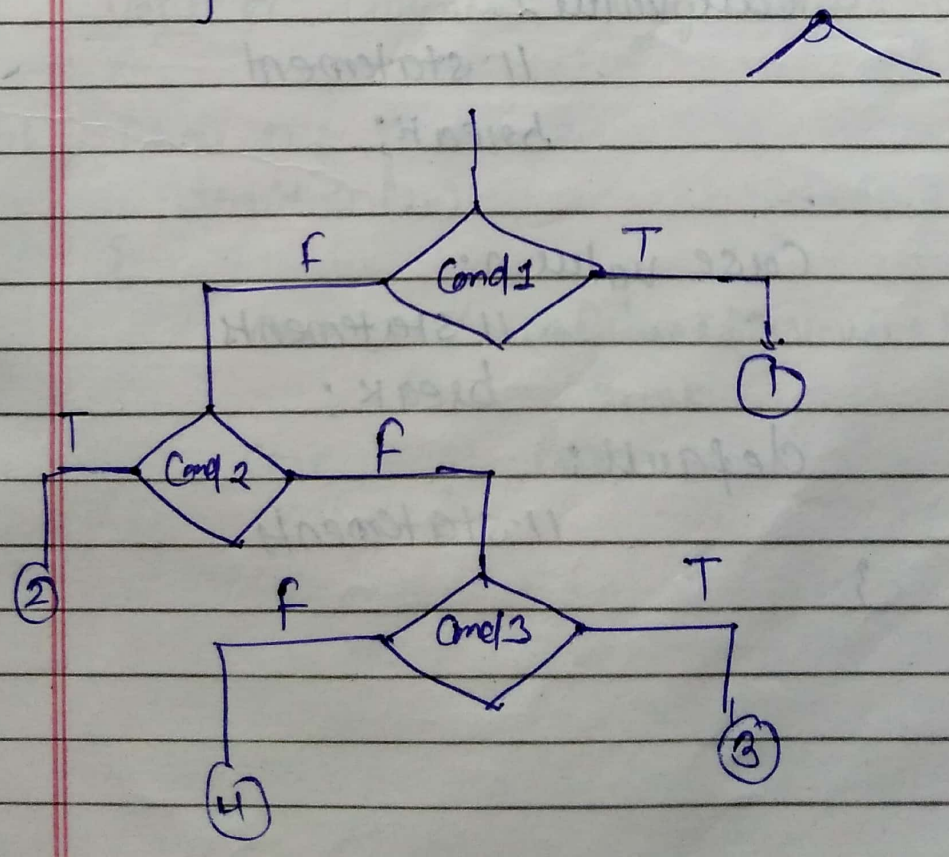
- used to execute one code from multiple conditions

- Also called multipath decision statement

DATE: / 20
PAGE No

- Syntax.

```
if(condition)
{
    // statement 1
}
else if ( condition 2)
{
    // statement 2
}
else if ( condition 3)
{
    // statement 3
}
else
{
    // statement 4
}
```



DATE : / 20
PAGE No

* Switch statement :-

- Acts as a suitable for long if-else-if ladder that is used to test a list of cases.

- When the expression match to a case then the associated statement with that case would be executed

- Syntax

```
Switch (expression)
{
    Case value 1 :
                // Statement
                break;
    Case value 2 :
                // statement
                break;
    ..
    Case value n :
                // statements
                break ;
    default :
                // statements
}
```

DATE: / 20
PAGE No

Ex:
```
int n; // 1 to 7
switch (n)
{
    case 1: Monday
            break;
    case 2: Tuesday
            break;
    :
    :
    case 7: Sunday
            break;
    default: Invalid Input
}
```

Q. WAP to check weather a character is vowel or Consonent using switch statement.

Sol
```
Char n; // A to z.
switch (n)
{
    case a: printf("Vowel")
            Break
    case e;
```

DATE : / 20
PAGE No

Sol"
```
char ch; // a to z
switch(ch)
{
    case 'a': // vovel
              Break;
    Case 'e': // vovel
              Break;
    Case 'i': // vovel
              Break;
    Case 'o': // vovel
              Break;
    Case 'u'; // vovel
              Break;
    ~~case~~ default ; // consonent

}.
```

* Iteration statement :

- Repeatedly executes a statement, known as the loop body, until the condition is false.
- Also called loop
- Consists of three parts

DATE : / 20
PAGE No

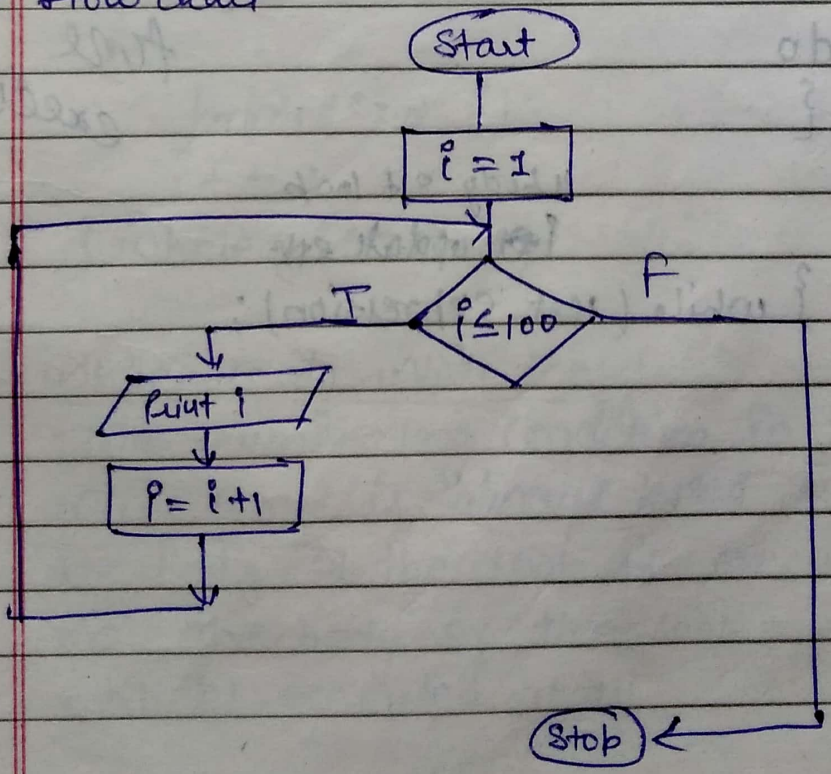
- initialization
- Test expression or Condition.
- Increment / Decrement or update value

- Types
- for loop
- while loop
- do-while loop

* Algorithm to Print 1 to 100 no.

(1) Start
(2) initialize i=1
(3) Repeat steps 4 & 5 until i<=100
(4) Print i
(5) i = i+1
(6) Exit

* Flow chart



DATE : / 20
PAGE No

* for loop :

- Syntax :

```
for (initialization; test expression; update exp)
{
    // body of loop
}
```

Initialization
while (cond)
{ body
inc/dec
}

* while loop

- syntax :

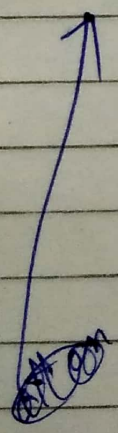
```
while (test expression)
{
    // body of loop
}
```

* do-while loop

- Syntax :

attleast one time execute

```
do
{
    // body of loop
    update exp
} while (test expression);
```



DATE : / 20
PAGE No

for int i

Ex
```
for (i=1, i<=100 ; i++)
{
    printF("%d", i);
}
```

- for while

=
```
int i=1
while (i<=100)
{
    printf("%d", i);
    i++;
}
```

For do while loop

```
int i=1
do
{
    printf("%d", i);
    i++;
} while (i<=100);
```

* while vs do while loop:
The terminating condition in the while loops is always tested before the body of the loop is executed. So, the body of the loop may not be executed at all

DATE: / 20
PAGE No

- In the do-while loop, the statement body is always executed at least once as the condition is tested at the end of the body of the loop.

* Break and Continue:

- While a break statement is encountered inside a loop, it will immediately terminated and execution resumes at the next statement following the loop.

- The continue statement terminates the current iteration of a loop and resumes execution back at the beginning of the loop body with the next iteration.

Ex
```
for (i=1, j=1 ; i<4 , i++, j++)
{
    printf("%d %d", i, j)
    if (i==j)
    {
      break
    }
}
```

DATE: / 20
PAGE No.

Q. int i, n;
int sum=0;
for (i=1 ; i<=n; i++)
{
sum = sum + i
}

→ for find the factorial

* int fact i;
for (i=n, i>=1; i--)
{
fact = fact * i ;
}

Q. 0 → 0
1 → 1
11 → 0 + 1

for (i=3; i<=n; i++)
{
f2 = f1 + f0
Printf("%d", f2);
f0 = f1 ;
f1 = f2 ;
}

DATE : / 20
PAGE No

Q Input any number and check the no. is Prime no. or not.

Soln

```
for ( i = 2, i <= n/2 ; i++)
{
    if (n % i = 0)
    {not Prime
     Break }
else }
    Prime
```

```
if ( i == (n/2 + 1))
  Prime
else
    Not Prime
```

*

```
*
* *
* * *
* * * *
```

i	j
1	1
2	2
3	3
4	4

```
For(i=1 ; i<=4, i++)
{
  for ( j=1, j<=i ; j++)
  {
    Printf("*");
  }

  Printf("\n");
}
```