C programming

Fundamental data types

Data type	Bytes	min val	Max val
Char	1	-128	127
int	2	-2^31	(2^31) -1
float	4	(6 digit after point)	

Derived

short int	2	-2^15	(2^15)-1
long int	4	-2^31	(2^31)-1
double float	8	12 digit after point	

Defining (syntax)

[data type] [variable name];

eg: Char a;

Constructs

- Conditional
- loop

Operators

==	equals
!=	not equal.
<	less than
>	greater than.
<=	less than equals
>=	greater than equals

Data loss

Suppose.

```
float a = 6.536;
int b = a;
Printf("%d", b);
Printf("%f", a);
```

o/p

6. Here .536 is lost.

6.536

When a float is made as int. value of decimal digits are lost

if... else.

Syntax

```
if (Condition)
{
 body;
}
else
 {
  body;
 }
```

eg:-

```
if (a<10)
{
 Printf ("a is less than 10");
}
else
{
 Printf ("a is greater than 10");
```

if else.

Syntax

```
 if (Condition)
{
 body;
}
else if (Condition)
{
 body
}
else
 {
  body
 }
```

Nested if, else

Note:- every if must have a matching else.

```
if (condition)
{
 body;
}
{ if (Condition)
 {
  body;
 }
 else (condition.
 {body;
 }
}
else
 {body
 }
```

Switch Case Construct

Syntax

```
Switch (condition)
{
Case 1:
    body;
    break; // break is used to terminate loop/condition
Case 2:
    body;
    break;
default:
        ↳ //default executes when no condition matches. It is optional. Break is not needed.
}
```

eg.

```
int a=1;
Switch (a)
{
Case 1:
   Printf ("one \n");
   break;
Case 2:
   Printf ("Two \n");
   break
default:
   Printf ("Invalid");
};
```

while loop.

Syntax

```
while (Condition)
{
  body;
  increment or decrement;
};
```

↳ Condition is Checked and body is executed

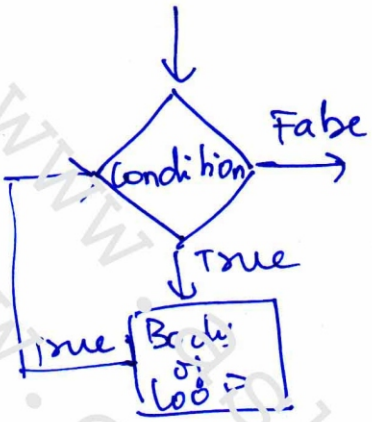
do while

Syntax

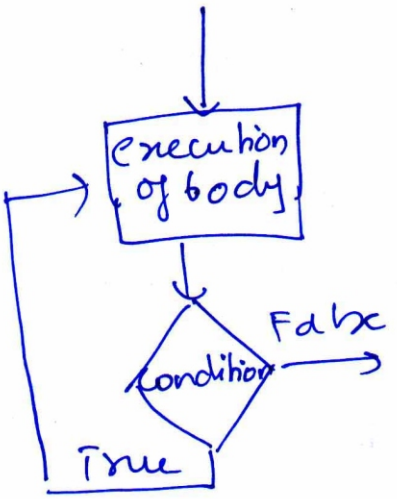
```
do
{
 body; increment or decrement;
}
while (condition);
```

↳ body is executed once then condition is checked

while



do while



for loop

Syntax

```
for (initialisation; condition; increment/decrement)
{
  body of loop;
}
```

eg

```
for (int i=0; i<=5; i++)
{
  printf ("i is %d", i);
}
```

Operators in C

logical

OR	||
AND	&&
NOT	!

unary (single operator)

int a=-12;

Binary operators

(when two operands are used)

+, -, *, %, /

eg
int x = y+z; → y, z: operands; + : operator

() parenthesis can be used to classify operation
eg :- (a+b)*(a-b).

Special operator (or) Ternary operator

```
if (condition)
{
    Statement;
}
else
{
    Statement
}
```

⇒ Can be simplified to.

eg:
x = (a>b) ? a : b;
If true a is printed.
If false b is printed.

Syntax

(Expression)? True : false;

Increment or Decrement

x = y++; is same as x = y+1;

x = y--; is same as x = y-1;

Types

* Pre increment
* Post increment

Pre increment

a = ++b;

is equal to.

a = b; b = b + 1

Pre increment

a = ++b;

is equal to.

b = b + 1; then a = b;

Post increment

a = b++;

is equal to.

a = b; b = b + 1.

Formatted output:

%d - decimal
%x - hexadecimal.
%s - string
%c - char.
%f - float

\n - new line
\t - tabspace.

Arrays.

Arrays are homogeneous elements in linear memory spaces.

Types

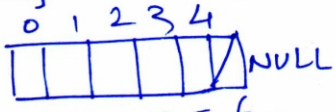
↳ one dimensional
↳ Multi dimensional.

eg:- Single - dimension

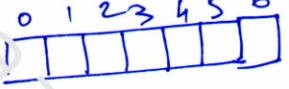
datatype Varname [array length];

array length starts from 0 and end with a null for char array

eg: char a [5]



int a [5]



Initialisation can also be done by accepting values

```
char a [i];
int i;
scanf ("%d", &i);
```

Two dimensional:

char a[4][2];

first array specifies number of strings
second specifies length of string (max limit).

∴ char a[2][4];

a = { Hi, Hello }

2 strings can be stored.

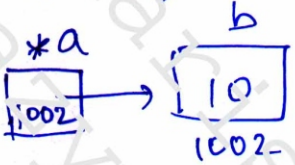
int a[2][2] are generally used for matrix calculations.

Pointer.

Pointer can be defined as variable used to store address of another element.

eg int b=10; *a;

a=&b;



Functions

Syntax

Returntype functionname (Arguments).

```
{
 body;
}
```

eg1:- main() is a fn.

eg2:- add (int a, int b).

```
{
  int c=a+b;
  return c;
}
```

Default return type is int.

#define is used to define user defined global variable

eg:- #define NULL 0

Structure

Syntax

```
Struct
{
  Data members;
} instance variable;
```

Union

```
Union
{
  Data members;
} instance variable;
```

Difference

	Structure	Union
i)	Instance can access multiple members simultaneously	i) instance can access only one member at a time.
ii)	memory is relatively more than union	ii) memory is max. value of available member.
iii)	used in implementation of nodal operation	iv) used in single access file operations

String manipulation

#include <String.h> is used

Standard fn

Strcmp() → StringCompare
Strcpy() → copies one string to another.
Strcat() → used to append a string with another.
Strlen() → finds string length

eg

x = Strcmp ("HI", "hi");
o/p
false (or) x = 0.

Strcat ("HI", "Hello");
o/p
HIHello

X = Strlen ("DCE");
o/p
3

file handling

fopen ⇒ opens file.

Modes.

r → read only
w → write only
a → append.
r+ → read + write
w+ → write + read.
a+ → Read + append.