Page No.
Date: 04/09/24

Foundation Of Computer Programming

Program : It is the Set of Instructions to perform some specific task.

WORA : Write Once Run Anywhere.

Machine works on Switching Theory.
ON : 1 OFF : 0

2009
Naman
Rohit
BCA

Data : Raw facts & figures.
Information : Meaningful things from the data / Processed data, over some Logics.
Eg., { 9, 2, 6, 5, 10 } data
in ascending Order : { 2, 5, 6, 9, 10 } Information

Procedure should be well-defined.
Algorithm is a Step by Step well-defined procedure & following are the characteristics :
↳ Well-defined ↳ I/P & O/P ↳ finiteness
↳ Unambiguous ↳ Flexible/feasible
↓ clear meaning ↓ alg. must solve every prbm of same type.

Procedure → Flowchart → Algorithm → Pseudocode → Program

Algorithm can be defined as finite set of steps to solve a Computational problem.
It has the following characteristics :

1. I/P & O/P : An alg. take some I/P & must produce at least 1 O/P.

2. **Well-Defined** - All the steps of an alg. must be define very well. It must be mentioned which one is the 1st step & which is the last step. It also describes which step will be Executed in what Condition.

3. **Finiteness** - An alg. must have a finite no. of steps. An alg. must terminate somewhere.

4. **Unambiguous** - All the steps must be written in clear language that means every step should has a single interpretation.

5. **Feasible** - It must be possible to execute every step of alg. for every input instance.

For Ex : Q: Write an algorithm to find out the largest element in given Sequence?

An array DATA[] of numeric values is in the memory. We have to find the LOC (location) & the value MAX of the largest element of a Sequence.

Step 1. [Initialization]
Set K = 0, MAX = DATA [0], LOC = 0;

Step 2. [Increment Counter]
K = K + 1;

Step 3. [test Counter]
if K > N-1, then write LOC, MAX & EXIT

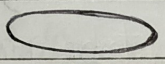
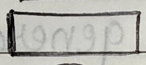
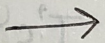
Step 4. [Compare & update]
if MAX < DATA [K] then Set LOC = K
MAX = DATA [K].

MAX -	1	19	6	9	20	8
LOC -	0	1	2	3	4	5

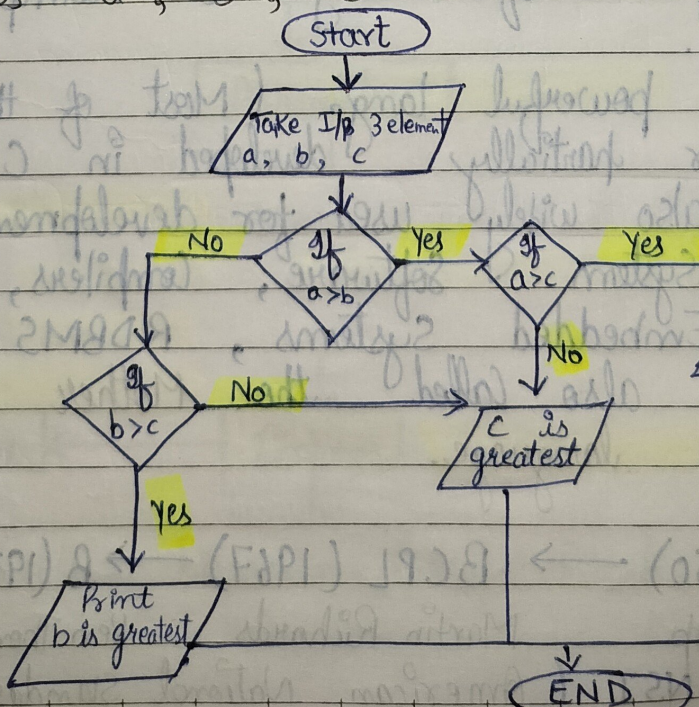
Step 5. [Repeat]
Go to Step 2.

* Flow Chart - Pictorial Representation of algorithm.

(Symbols for flow chart :-

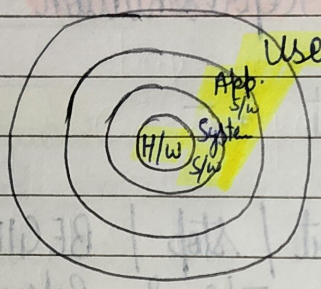
Oval		Start / Stop / BEGIN / END
Parallelogram		Take I/P & Represent O/P.
Rectangle		Processing / Process / Logic.
Circle		Connector
Arrow		Control flow / Data flow.
Diamond		Used for Decision making.

Q: By Using flow chart find the Largest among 3 Elements a, b, C.



AND operator

```
if ((a>b) && (a>c))
    Print a is greatest.
if ((b>a) && (b>c))
    b is greatest
if ((c>a) && (c>b))
    c is greatest
```



- System S/w : software that directly interact with H/w.
  Ex - O.S, Windows, Linux, Android

4/09/24

+ **C - language** : C is a general purpose Procedural programming language developed by Dennis - Ritchie at AT & T's Bell Labs in 1972.

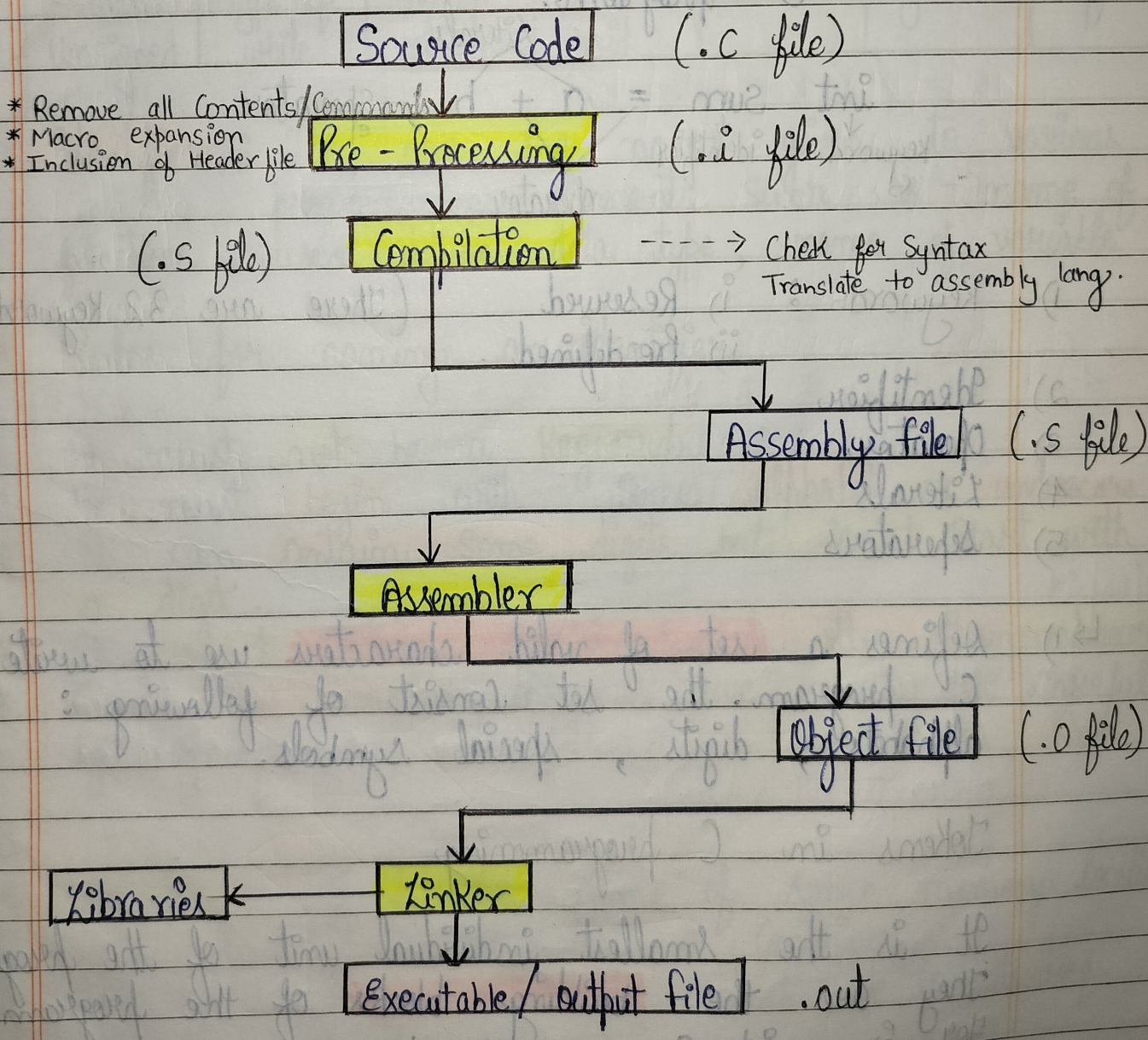
→ It is a High level language.
→ It is often refered as mid-level lang. because it supports to low level programming constructs.
→ The development of C begin to reimplement the UNIX OS.
→ It is a powerful lang. (Most of the OS are Completely or partially developed in C lang.).
→ It is also widely used for development of OS, Kernels, System Software, Compilers, Device drivers, Embedded systems, RDBMS, System S/w.
→ This is also called the Mother of all Programming language.

→ ALGOL (1960) → BCPL (1967) → B (1970) → C (1972)
International Grp — Martin Richards — Ken Thompson Dennis Ritchie
→ (1989) ANSI (American National Standard Institute).

Page No.:
Date: ___/___/____

+ **Compilation** – The process of translating the source code written in high level lang. to low level lang. (machine code) is known as compilation.
It is done by some special S/w called as **Compiler**. Compiler checks source code for any syntactical Error or any structural Errors & generates the object code with .o extension (in linux os) or (.obj in window).

* There are 4 stages of Compilation.

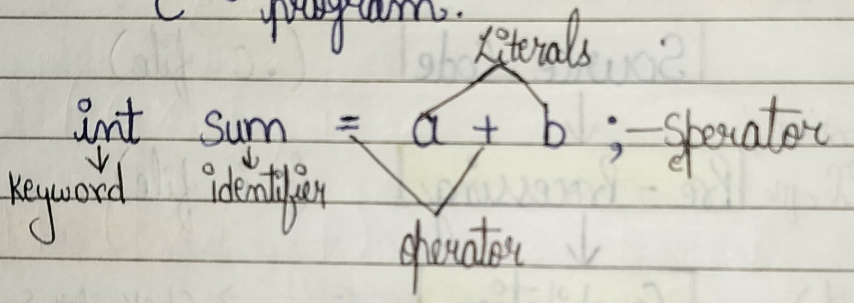


Page No.
Date 12 09 24

"C" Character set

→ Alphabets : (a-z) or (A-Z) case sensitive.
→ Digits : (0-9)
→ Special Symbols : #, *, @, +, -, %, \, &, ., ;, ', ,
→ White space, Blank Space.

→ Tokens : They are the smallest unit present in C program.



1) Keywords : i) Reserved (There are 32 Keywords)
ii) Pre-defined
2) Identifier
3) Operator
4) Literals
5) Seperators

↳ 1) Defines a set of valid characters use to write a C program. the set consist of following : alphabets, digits, special symbols.

Tokens in C programming

It is the smallest individual unit of the program. They are the building blocks of the program.
For Ex - int Sum = a + b.

13/09/2

Keyword : They are the reserved words whose meaning is pre-defined by the programming lang. specification. They convey some special meaning to program & can't be used for any other purpose.
There are total 32 keywords.

Auto	break	case	char	double	const	do
continue	default	else	int	long	register	return
enum	short	signed	sizeof	static	extern	struct
switch	typedef	union	float	for	goto	if
unsigned	while	volatile	void			

2÷ Identifier : They are the names given to various programming elements such as - name of function, user define data type, name of variable

↳ Rules for naming an identifier

1. It must not be a keyword
2. It must begin with some alphabet or underscore.
3. It can contain some digits but can't start with a digit.
4. It can't contain special symbols other than underscore.
5. C is case sensitive that is we must be careful by naming an identifier. Ex- num, NUM, Num.

13/09/24

- Variables - It is the name of some memory location
  It works like a container.
  It can vary its content a/c to its req.

Date: 18/09/24

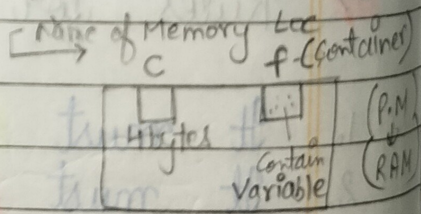
Q. Write a program to convert temp. given in degree Celcius into F.

```c
#include <stdio.h>          // header file; # include = Pre-processing directive
int main ()                 // (Entry pt. of any program)   int = (Keyword) return type; main = funⁿ
{
    int C, f;               // data type; (Declare the variables before using) / memory loc)
    printf("Enter the temperature in degree Celcius");
    *Scanf ("%d", &C);      // %d = format specifier
    f = (C * 9/5) + 32;     // left <- assign <- Right
    Print f ("%d", (f); return 0;
}
```

& – address of
* – value at
*(&f);

(f) → If there is & so it shows the address of f not result.

-:- **Identifier** : i) Varicable
ii) Constant
iii) function name



17/08/24

3) **Operators in C.** :- Operator is a symbol that perform some operation over operants. (values).

a + b ;
operants
↑ Operator

i) Arithemetic operator
ii) Assignment operator
iii) Relational operator
iv) Logical operator
v) Bitwise operator
vi) Conditional operator (Ternary operator).
vii) Increment / Decrement operator
viii) Other (., →, *, &, size of).

Uniary { of the no. of operants is single
Binary, Ternary operator

i) Arithmetic operator are used to perform basic arithemetic operations.
(+, −, *, /, %)
They are in the category of Binary operator. Or it need at least 2 operants. (modulous operator)
Eg., 2+3=5, 2/3=0, 2%3=2.

ii)* Assignment operator is used to assign a value to a variable. the values assign from right to left side. (a=b) a=4 b=5 ∴ a=b a=5, b=4.

iii) Relational operator (==) is used to check relation b/w two operants. Binary operator.
Ex- • > greater than, a>b (a=5, b=8).
• < less than, a<b
• <= a==b. (a=2, b=2)
• >= a≤b (b=5). (1,2,3,4,5)
• (≠) a!=b. • =

iv) Logical operator are used to combine two or more relation logically.
There are 3 types of logical operator.

↳ i) AND operator (&&) It works over two operator.

Truth Table :-

A	B	A&&B
f(0)	f(0)	f(0)
0	1	0
1	0	0
1	1	1

both Cond^n should be true.

↳ ii) OR operator (||)

A	B	A||B
0	0	0
0	1	1
1	0	1
1	1	1

It works over binary operator.

Either of 1 or both should be true.

↳ iii) NOT operator / Inverter / (!)

A	!A
1	0
0	1

It works over Uniary operator.

v)* Bitwise operator

Byte is the smallest unit of memory.

$(5)_{10} = (101)_2$

$= (00000101)$

1 byte = 8 bits

1024 byte = 1 KB

1024 K byte = 1 MB

1024 MB = 1 GB

$(23)_{10} = (10111)_2 = (00010111)$

Short Trick ↓

{N... 64, 32, 16, 8, 4, 2, 1}

Page No.:
Date: __/__/____

Bitwise operators perform operations on binary bits.
Let's suppose - a = 10 (00001010) b = 5 (00000101)
we collect data from bottom & before digits, add 0 to make it 8 bits atleast

* ↳ & : Bitwise AND performs AND operations on 2 binary bits when both input is 1 then result is 1 otherwise 0.

a = 10, b = 5

```
  a = 00001010
& b = 00000101
  ------------
    00000000
```

if [10 & 5] = 0 [f]
[10 && 5] = 1 [T] → if we write value other than 0 i.e. its a true value.
Logical AND (8&&0 = f)

```
#include <stdio.h>
int main ()
{
int a = 10, b = 5;
if (a & b) → check the value
{
printf ("Bitwise operators");
}
if (a && b)
{
printf ("Logical AND operators");
}
return 0;
}
```

↳ Bitwise OR returns 1, if any of the two binary bits are 1 otherwise 0.

a = 10, b = 5
(00001010) | (00000101) = 00001111
↳ OR
8 4 2 1
(15)10

(10 | 5) = (15)10
Value other than 0 So it will Executed.

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

Page No:
Date 23/09/24

↳ XOR (^)

Exclusive OR : If the bits are different then O/P is 1 otherwise 0.

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

Ex- a = 4 = 00000100
b = 1 = 00000001
00000101
4 2 1

(5)2 . Ans

a = 7 = 00000111
b = 4 = 00000100
a ^ b = 00000011 → 3. Ans

a | b = 00000111 → 7. Ans

↳ Bitwise NOT (~) / Compliment Operator

Most Significant Bit indicates Sign: 0: +ve, 1: -ve

a = 4 = 0100
~a = 1011
MSB → -3 Ans

Q: Swapping 2 nos. using bitwise operator.

```
# include <stdio.h>
int main ()
{
 int a, b;
 a = 4, b = 3;
 a = a^b;
 b = a^b;
 a = a^b;
printf ("%d %d", a, b);
}
```

```
a = 4  =  0100
b = 3  =  0011
          ----
          0111        ans → 7
```

```
a = 7  =  0111
b = 3  =  0011
          ----
          0100        b = 4
```

```
b = 4  =  0100
a = 7  =  0111        a = 3
          ----
          0011
```

25/09/24

↳ Bitwise Left shift operator (<<)

a = 15 << 1

a = MSB 0000111(1) Least SB = $(15)_{10}$

- It requires only two operants.

a << (1) This indicate how many bits to shift.

a << 1, 00011110 = $(30)_{10}$
a << 2, 00111100 = $(60)_{10}$
a << 3, 01111000 = $(120)_{10}$

} By process

a << 4 = 15×2^4 = 240
a << 3 = 15×2^3 = 120
a << 2 = 15×2^2 = 60

} for checking the ans.

NOTE: Bitwise: &, |, ^, ~, <<, >> *important

↳ Bitwise Right shift operator (>>).

a = 15 >> 1
a = 00001111

a >> 1, 00000111 = $(7)_{10}$
a >> 2, 00000011 = $(3)_{10}$
a >> 3, 00000001 = $(1)_{10}$
a >> 4, 00000000 = $(0)_{10}$

} By process

a >> 1, 15/2 = 7
a >> 2, $15/2^2$ = 3
a >> 3, $15/2^3$ = 1
a >> 4, $15/2^4$ = 0

} for checking the ans.

vi) Increment / Decrement operator

a++ (after the variable : Post fix)
a = a+1;

a--
a = a-1;

(always ↑se or ↓se by 1).

↳ Applicable only on the variables NOT for Constant & Expression.

a++ (post fix) , ++a (pre fix).

↳ Unary operators that ↑ses or ↓ses the value of a variable ↓ gradually by 1.

- ```
  #include <stdio.h>
  int main ()
  {
  int a = 5, b, c;
      b = a++;
      c = ++a;
  printf ("%d %d", a, b, c);
  }
  ```

5 ⟶ 6 ; 7 ←

Ans b = a ++ ; a = ++ a

∴ a = 7 (updated value), b = 5, c = 7

- a = 5
  b = ++a ;
  c = a++ ;
  a, b, c = ?

b = (++a) = 6 ; a = a+1

c = a++ ; 6 → 7

∴ a = 7, b = 6, c = 6.

27/09/24

```
#include <stdio.h>
int main ()
{
  Char var = 3;
  printf ("%d", var<<1);
  return 0;
}
```

discard (LSO)

a = 3 = 00000011

a<<1 = 00000110 ⇒ 6 $(3 \times 2^1 = 6)$

4 2 1

a<<3 = 00011000 ⇒ 24 $(3 \times 2^3 = 24)$

16 8 4 2 1
```

vii) **Conditional Operator** (Ternary Operator)

↓

3 no. of operants.

< Condition > ? < True statement : false statement ;

```
a = 10    (a>b) ? printf ("a is greater"): printf
b = 5                    ("b is greater");
```

↳ Simple we can say that Conditional operator is the Replacement of Simple if & Else block.

viii) **Others**

- & (...) address off / referency off.
- * (address) value at (...). / dereferency.
- → (...) Access the structure member.
- . (...) Access the member of object.
- .* sizeof (..) Eg., int a;

  → (Size of (a);

Operators	Type
++ , --	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Rational operator
&&, ||, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	conditional or Ternary operator

Binary operator

Page No.:
Date: 30/9/24

Q:
```
#include <stdio.h>
int main()
{
int a = 20, b = 15, c = 10, d = 5, e;
e = a + b * c/d ;
printf ("%d", e);
return 0;
}
```

we have to check the **precedence order**

a + b * c/d

20 + 15 * 10/5 = 20 + 150/5 = 20 + 30 = 50

(a + b) * c/d ;

35 * 10/5 = 350/5 = 70.

(a+b) * (c/d);

(35) * (2) = 70.

↳ Pre : Process first then assign value
↳ Post : Assign first then increment

Q:
```
#include <stdio.h>
int main()
{
int x = 5, y = 5, z ;
printf ("%d", x);
z = x ++ ;
printf ("x = %d and z = %d", x, z);
z = ++ y ;
printf ("z = %d and y = %d", z, y);
return 0;
}
```

z = x++ (Post)

assign first, 5 + 1 = 6

z = 5

z = ++y (Pre)

z = 6, y = 6

assign after that

↳ Data Type

Each variable in C has an associated data type. It specifies the type of data that the variable can store like integer, character, floating point, double etc. Each data type requires different amount of memory & has some specific operations which can be performed over it. Different data type also have different ranges upto which they can store the numbers. These stages may vary from Compiler to Compiler on 32 bit GCC Compiler.

Data Type	Size (bytes)	Range	format Specifier
short int, short	2	-32,768 to +32,767	%hd
Unsigned short int	2	0 to 65, 535	%hu
Unsigned int	4	0 to 4294, 967, 295	%u
int	4	-2, 147, 483, 648 to +2, 147, 483, 647.	%d
long int	4	-2, 147, 483, 648 to +2, 147, 483, 647.	%ld
Unsigned long int	4	0 to 4, 294, 967, 295	%lu
long long int	8	-(2^63) to +(2^63)-1	%lld
Unsigned long long int	8	0 to 18, 446, 744, 0737, 09, 551, 615.	%llu
Signed char/char	1	-128 to +127.	%c
Unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38.	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Page No.:
Date: 03 / 10 / 24

①
```
#include <stdio.h>
int main()
{
Signed char a = 125;
    a = a + 10; // 135
printf("%d", a);   -121
printf("%d", a);
return 0;
}
```

Range - -128 to +127
-128, -127, -126, -125,
-124, -123, -122, (-121),
+125, +126, +127

②
```
#include <stdio.h>
int main()
{
char a + 127;
    a = a + 10;
printf("%d", a);
printf("%d", a);
return 0;
}
```

③
```
Unsigned char a = 125;
    a = a + 10;
    //135.
```

④
```
char a = 336;
printf(....)
```
336 - 256 = 80
//80
P will be printed.

Q: Size of int(4), Size of char(1), Size of short int(2), Size of double

```
#include<stdio.h>
int main()
{
printf("Size of int %d", Size of (int));
printf("Size of char %d", Size of (char));
printf("Size of short int %hd", Size of (Short int));
printf("Size of double %lf", Size of (double));
return 0;
}
```

Ranges of all the data types

%d	to	%d
INT_MIN		INT_MAX
LONG_MIN		LONG_MAX
LONG_LONG_MIN		LONG_LONG_MAX

for range, we have to include one more header file

```
#include <limits.h>
```