Date

Placement Scenario for Technique

Capgemini

	Analyst	mid Analyst	Senior Analysist
	No need code	no need code	Need coding
Salary	4 LPA	05.6 LP	7.5 LPA
inhand	27,721 / month	41,000 / month	55,813 / m
		38k-41k	① Mid coding
			② Mid + coding round

Cg → senior → mid coding, mid+ coding

① English
② Gaming ⑥–④ 6min/game
③ Snippest (psudocode)
④ Working Questions

Snippest – 3 to 16 line code
where, predict the o/p.
TCS → Correct the code
Infosys → Change the code.

TCS

TCS Ninja	TCS Digital
- 3.5 L	- 7.5 L
26,000 / m	- 48k - 53k

TCS exam – 3600 mark

Free (on campus) NQT — Paid (999 rs) NQT

NQT – National Qulify Test

SASS-

GoodLuck | Page No.
Date

TCS - 3600 marks

1800 apti

1800 → Advanced aptit

1200 apt

600 → coding

300 15min

300 30min

WIPRO

Wipro delight
3.5 LPA
23 K

Wipro Turbo
6.5 LPA
44 K.

TCS

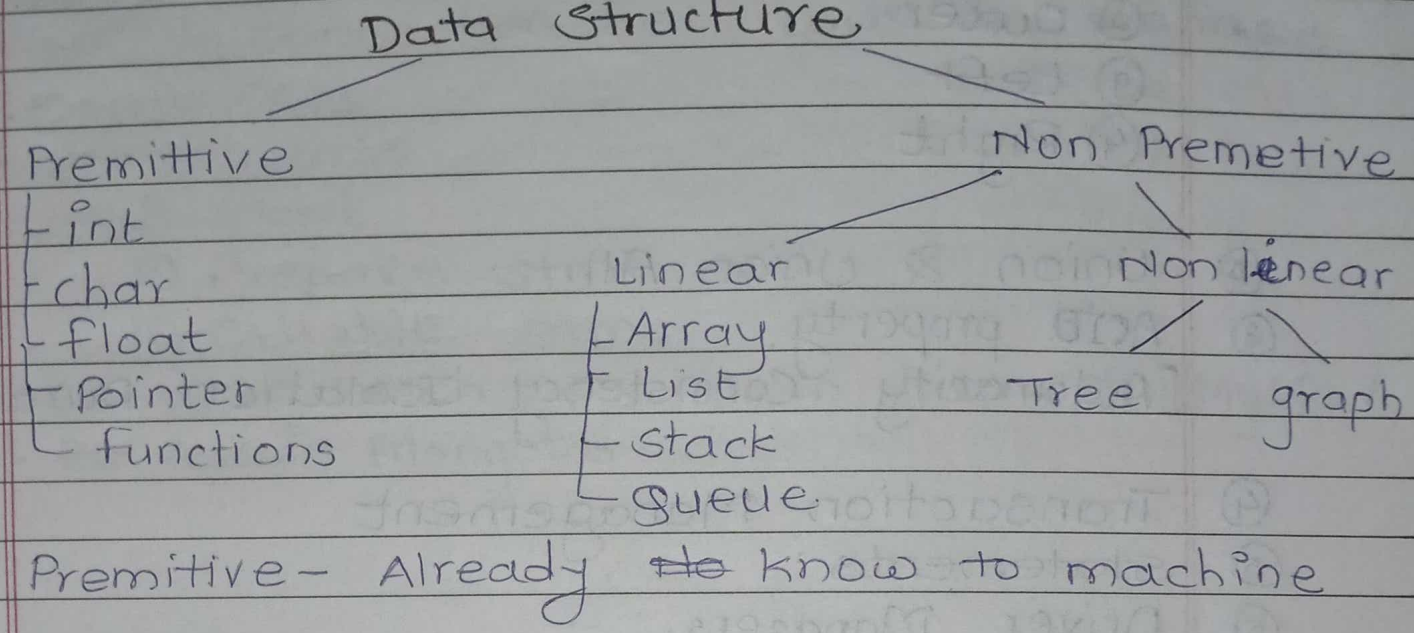
Ninja → Digital → Inovator
3.36 L 7.002 L 12 LPA

Wing 1 → Tech + (coding + Snippet)
Business
Logical

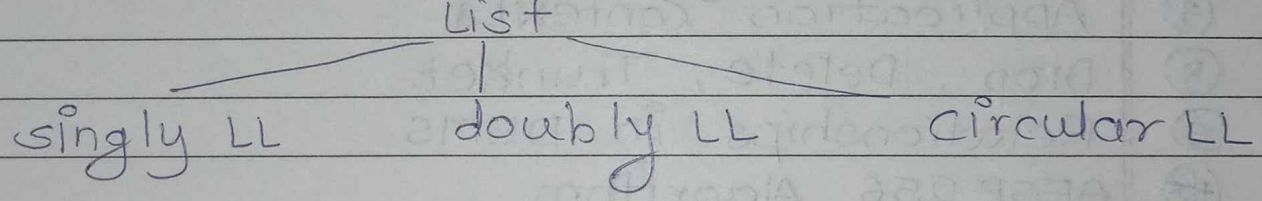
GoodLuck | Page No. | Date

Syllabus - for Technical.

Data structure & object Oriented programing system



Premitive - Already ~~to~~ know to machine



Sorting Techinques

1. Bubble sort
2. selection sort
3. Insertion sort
4. Quick sort
5. Merge sort
6. Heap sort
7. Radic sort
8. Bucket sort

GoodLuck | Page No. | Date

DBMS

① Joint
- ① Inner
- ② Outer
- ③ Left
- ④ Right

② Union & Union All

③ Acid property
[Automacity Consistency Isolation Durability]

④ Transaction Management

⑤ Statements

⑥ Driver Managers.

⑦ Application Contexts.

⑧ Drop, Delete, Trunket

⑨ Relationships in DBMS

~~#~~ AESP 256 Algorithm
└ Encrepption of data & Decription of data from decripted data]

~~##~~ Cloud.
└ services
- implementation
- Application

* Certification
① AIML
② DL
③ Data Structure
④ AWS, CC, AZ900, (Asure 900), GCDL
⑤ WD

GoodLuck | Page No. | Date

Core Java

- Advanced Java.
- JDBC - Java Database Connectivity
  ↳ connection bet? code & database
- Containers
- Statements
  - ① stmt
  - ② Prepare stmt
  - ③ Callable stmt
- ~~Oracle~~ Work Bench / PG Admin / No SQL / MySQL /
- ~~PG Admin~~ MongoDB

C programming

Datatypes :-

	Premetive	Non-Premetive	User-defined
1	integer	Array	Struct
2	float		Union
3	long		enumeration
4	double		class
5	char		

1. int - 2 byte - %d and also %i

2. float - 4 byte - %f
   It can store upto 6 decimal point.
   e.g. a = 3.14
   : a = 3.140000

3. Double :- 8 byte %lf
   upto 10-12 decimal point

4. long - 4 byte - %ld

5. Char - 1 byte / 8 bits , %c

WAP to write a-z alphabate without using ASCII value.

```
for (char i='a'; i<='z'; i++)
{
    printf("%c",i)
}
```

```
int x=10
if (x==10)
  printf ("Yes");
  printf ("No");
```

O/p - Yes.
जर आपण curly braces दिले नाहीत तर if condition नंतरची फक्त line print होते.

Ternery operator.

condition ? exp1 : exp2 ;

* printing Hello World without using semicolon

```
if (printf ("Hello World")
{

}
```

* Operator precidence.
  i) !
  ii) *, /, %
  iii) +, -
  iv) <, <=, >, >=
  v) ==, !=
  vi) &&
  vii) =

a = 10, b = 11

c = a++ + --b = 20

Loops

① For loop-

```
for ( initialisat^n ; condition ; ++/--)
{

}
```

eg.
```
for(int i=0; i<10 ; i++)
  {
    printf ("%d", i);
  }
```

② While

```
while (condition)
{
  ++/--
  stmt
}
```

③ do-while.

```
do
{ ++/--
  stmt
} while (condition);
```

* * * *
* * * *
* * * *
* * * *

```
for (int i=0 ; i<4 ; i++)
{
  for (int j=0; j<4; j++)
  {
    pf ("*");
  }
  pf("\n");
}
```

*
* *
* * *
* * * *

```
for (int i=0; i<4; i++)
{
  for (int j=0; j<=i; j++)
  {
    print ("*");
  } pf ("\n")
}
```

- WAP to print alternate no / odd no upto 1000

```
int i=1
while(i<=1000)
{
  printf ("%d", i)
  i+2
}
```

- To print a table 12. using do-while loop

```
i=1
do
{
a=12*i;
 print (a);
 i++;
} while (i<=10);
```

27/01/24

GoodLuck | Page No. | Date

- Switch Case

case 1 : given no is even or not
case 2 : given no is odd.

```
int no;a;
pf("Enter a no");
sf("%d", &no);
pf("Enter 1: Even , 2: odd");  sf("%d", &a)
switch (a)
{ case 1: if (no % 2 == 0)
          {
            pf("no is even")
          }
          break;

  case 2: if (no % 2 != 0)
          {
            pf("no is odd");
          }
          break;

  default: pf("Enter proper no");

}
```

- Array

Syntax - declaration.
int name_of_array [size_of_array]

e.g -
int a[5]

GoodLuck | Page No. | Date

Array initialization -
int arr[5] = {2, 3, 4, 5, 7};

- [If we only declare an array and try to print out its value then the garbage value will be print]

- int arr[3] = {0}; o/p |0|0|0| arr
int arr[3] = {0,0,0};

Q. WAP to take i/p to array.

```
int arr[5];
for (int i=0; i<5; i++)
{
  sf("%d", &arr[i]);
}
for (int j=0; j<5; j++)
{
  pf("%d", arr[i]);
}
```

Q. i/p arr[10] = {2, 111, 8, 14, 7, 123, 111, 9, 99, 333};

```
for (int i=0; i<10; i++)
{
    if (arr[i] == 99)
    {
        pf("Present");
        pf("index %d", i); break;
    }
}} else
    { pf("Not present"); }
```

Q. how to find smallest no from given array

arr[10] = {2, 111, 8, 14, 7, 123, 101, 9, 99, 333}

```
for (int i=0; i<10; i++)
{ int temp = arr[i]
   for (int j= i+1; j<10; j++)
   {
      if ( arr[i] < arr[j] )
      {
        pf ("%d", i)
      }
      el
```

```
int min = a[0]
for (i=0 ; i<10 ; i++)
{
   if (a[i] < min)
   {
     min = arr[i];
   }
}
pf("smallest element is %d" mid);
```

Q How to find the size of array

1. sizeof (arr[10])

Q length of array using sizeof () funct

len = sizeof (arr) / sizeof (arr[0])

- Symbolic Constant

```
#define A 10
```

```
#define COL 10;
#define ROW 10;
 int arr [ROW] [COL]
```

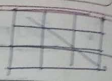
- Multi Dimentional Array (2D Array)

2, 111, 8, 14, 7, 123, 101, 1, 99

```
int arr [3] [3]
for (int i=0; i<3; i++)
{
   for (int j=0; j<3; j++)
   {
      sf ("%d", &arr[i][j]);
   }
}
```

```
// sum of all element
   for( i=0 ; i< ROW ; i++)
   {
    for( j=0; j< COL ; j++)
    {
    int c [i] [j] = c[i] [j] + arr[i][j]
    }
   }
     printf ("%d", c[i][j]);
```

Sum of diagonal element



```
for(i=0;i<3;i++)
{
  for(j=0;j<3;j++)
  {
    if(i==j)
    {
      sum+=arr[i][j]
    }
  }
}
```

Sum of both diagonal element

```
for(i=0;i<3;i++)
{ for(j=0;j<3;j++)
  {
    if(i==j || i==0&&j==2 || i==2&&j==0)
    {
      sum+=arr[i][j]
    }
  }
}
```

OR

```
for(i==j || (i+j)==2)
```

1 2 3
4 5 6
7 8 9

String

library - #include<string.h>

converting lowercase to uppercase

```
char alpha[26] = {a - - - - - z};
  for(i=0 ; i<26 ;i++)
  {
    alpha[i] = alpha[i] - 32
  }
  for(i=0; i<26 ; i++)
  {
    printf("%c", i)
  }
```

function.

① strlen() ② strcpy() ③ strcmp()
④ strcat() ⑤ strlwr() ⑥ strupr()

① int len = strlen(alph)

② char str[100]

```
char str[] = "HelloTeam";
  strcpy(str, str1)
         destination source
        OR
  strcpy(str, "chinya");
```

③ strcmp - compare

```
strcmp(str, str1)
```

④ strlwr - strlwr(alpha) OR strlwr("HELLOW")

strupr - ——"—— ——"——

GoodLuck | Page No. | Date

⑥ strcat (str1, str2)

By Rehan Sir

Increment - Decrement

Preincrement : ++i
Postincrement : i++
Predecrement : --i
Postdecrement : i--

a = 3	a = 3	a = 3	a = 3
b = a++	b = a--	b = ++a	b = --a
o/p a = 4	a = 2	a = 4	a = 2
b = 3	b = 3	b = 4	b = 2

Q. j = 11 o/p.
i = j++ i = 11 j = 12
j = 1 j = 11
j = i++ j = 11 i = 12

Q. p = 9
p = p+1
p = p+1+2
i = p++
i = i+p+1
i = i++

p = 9
p = 10
p = 13
i = 13 p = 14
i = 28
i = 29

p = 9 10 13 p 14
(1) (13)
i = 13 + 14 + 1
= 28
i = (29)

syntax rule. follow any

Snippets.

GoodLuck | Page No. | Date

Q. set b = 10
for (each a from 1 to 4)
b = b + a
c = b/5
c = 20/5
c = 4

10 + 1 + 2 + 3 + 4
b = 20

Q. i/p a = 2 & b = 4
8 2
int Fun (int a, int b)
{ n = 0
if (b < 1)
return n.
else
return fun (a+b+2, b-2)
}
8, 2
8 + 2 + 2
12, 0
n = 0

Q. int a, b, c, d.
a = 14, b = 15, c = 16
if (a > 6)
b = c - a b = 16 - 14 = 2
if a > c 14 > 16
d = b + c d = 2 + 16 = 18
else
d = b - c d = 2 - 16 = -14
else
d = a + b + c - 3
print d.

Q a = 56, b = 876

```
fun mul (a,b)
    t = 0
    while (b != 0)
        t = t + a
        b = b - 1
    End
    return t.
End.
```

t = 49056

Q if m = 9, n = 6
m = m + 1 → m = 10
n = n + 1 → n = 7
m = m + n → m = 10 + 5 = 15
if (m > n) → 15 > 5
print m → m = 15
else
print n

Q int n
set n = 11
int arr [4] = [10, 11, 1, 12]
arr [1] = arr [1] - arr [3]
[2] arr [2] + arr [2]
[3] = 1 + arr [3]
n = arr [3]
if (n > arr [2]) → 13 > 2
n = n - 1 → n = 12
else
n = n - - 1

Q int x, y, z
x = 9, y = 2, z = 1
if (x + y > 8) → 9 + 2 = 11 > 8
if (7 > x + z) → 7 > 10
End if

if (9 > x - z) → 9 - 1 = 8 < 8
z = 9
end if
y = y + z + 1
2 + 9 + 1 = 12
end if

print (x + y + z)
9 + 12 + 9 = 30

Q. char [] text = TESTSTRING
int a, c
char ch = T
c = 0
for (each a from 0 to length of text) → 10
if (text [a] == ch) → 3
c = c + 1
end
end
if (c > 0)
printC
else
print 0

c = 3

Q. t=6, h=9 x=0
int c
if (h>t) 9>6
for (c=t ; c<h = c=c+1)
6 9
x = x+c 6+7+8 = 21
End
print x

x = 21

Q. do
int i=3
print i+3 6, 5, 4
i = i-1
while (i != 0)

Q. a=30, b=60, c=90
int a, b, c, sum
sum = a+b+c 30+60+90=180

if (sum==180) & (a != 0) & (b!=0)
(c != 0))
print "success"
else
print "Fail"
end if

o/p = success

Q. a=2, b=6
int fun (int a, b)
if (a>0)
if (b>c)
return a+b+fun(a+1,0)+fun(a+2,0)
+fun(a+3,0)
end
end
return a+b
end

o/p = return 2+6+

Q. set p=3, q=4, r=5
p = p+p p = 6
p = (q+10)+p p = 25
r = r+2 r = 7

if ((p-q)< (q+p)) 21 < 29
q = 10^r
end if

print q+p+r = ~~10^7 + 25 + 7~~
13+25+7 = 45 ~~10^7 + 32~~

o/p = 7 = 0111
10 = 1010
1101 = 13

13 ^ 2
1101
0010
1111 = 15

78
64
14

- 78 binary conversion
  64 32 16 8 4 2 1
  1 0 0 0 1 0 0

78 =

241-128 = 13

- 241
  128 64 32 16 8 4 2 1
  1 0 0 0 + + 0 +

241 = 11110001

- 888

2|888
2|444 -0
2|222 -0
2|111 -0
2|55 - 1
2|17 - 1
2|8 - 1
2|4 - 0
2|2 - 0
1 - 0

888 = 1000111000

- 7392

2|7392
2|3686-0
2|1843-0
2|921 -1
2|460-1
2|230-0
2|115-0
2|57-1
2|28-1
2|14-0
2|7-0
2|3-1
1-1

7392=1110011001100

by Pan

7/02/2024

Functions -

Ø. ~~int~~ - Functions

user defined Library / build in funct

declarat^n → #include
void A()
int main()
{
calling → A()
}
defination → void A()
{
}

Write a function that print Namste if user is indian Bajour if user is french

```
#include<stdio.h>
void A()
int main()
{ char c;
   printf("Enter your conty : I/F");
   scanf("%c",&c);
    A(c)
}
   void A(char c)
{  if ( c=='I' || c=='i')
   {  printf ("Namaste"); }
    if ( c=='F' || c=='f' )
   {  printf ("Bajour"); }
   else
   {  printf (" wrong ..."); }
}
```

Arguments are actual parameter which pass during function call.

Parameter are formal parameter which are pass during function defination.

e.g- int Add (int a, int b); — Actual param

int Add (int a, int b) — formal param
{
return a+b;
}

Q Calculate square of given no. by user using library function

```
#include <math.h>
#include <stdio.h>
void square (int no)
int main()
{ int no;
  printf("Enter a no");
  scanf("%d",&no);
  square (no);
}

void square (int no)
{
  printf("Square of given no is: %d",
                                pow(no,2));
}
```

When we pass any value to function then we pass it copy to the function.

Q Write a function to convert celcius to fahrenheit.

```
#include <stdio.h>
int void f(float cels)
{ float f;
  return f=c*(9/5.0)+32
}

int main()
{ float cels;
  printf("Enter celcius value:");
  scanf("%f",&cels);
  f(cels);
  printf("Fahrenheit: %f",f);
}
```

OR

```
int main()
{ float cel;
  pf("Enter:");
  sf("%f",&cel);
  pf("%f",(cel*(9.0/5.0))+32));
```

Q WAP to calcute percentage of 3 subject.

```
#include <stdio.h>

void percent (int a, int b, int c)
```

```
int main()
{ int m1, m2, m3;
  pf("Enter mark of 3 subject");
  sf("%d %d %d", &m1, &m2, &m3);

  percent(m1, m2, m3);
}

void percent(int a, int b, int c)
{ float avg;

  avg = ((a+b+c)/300) * 100;
  pf("Percentage: %f", avg);
}
```

Q. Palindrom -

```
char ch[5] = "madam";
j = 5
for(i=0; i<5, j>0; i++, j--)
{  if (ch[i] == char[j]
   {
      flag = 0
   }
   else
   {
      flag = 1;
      break;
   }
}

if (flag = 0)
{ pf("string is palidrom");
}
else
{ pf("string is not palindrom");
```

or

```
while (s!=e)
if
++,e--;
```

* Recursion :-

1) Anything that can be done with iteration can be done with recursion & vise versa
2) Recursion can be sometime more simple solution
3) ~~This~~ is base case is a condition that stop recursion
4) Iteration as infinite loop & recursion as stack overflow

Q. Sum of first natural no using recursion

```
int sum (int n)
{ if (n == 1)                    // base case
  {
     return 1;
  }
  int finalsum = sum (n-1) + n;
  return finalsum;
}
int main()
{ int no;
  pf ("Enter no");
  sf ("%d", &no);
  sum (no);

  pf ("sum %d", finalsum);
}
```

sum(6)
sum(5)+6 = 21
sum(4)+5 = 15
sum(3)+4 = 10
sum(2)+3 = 6
sum(1)+2 = 3
sum(1)

Q Write a recursive program to find the factorial of given no.

```
int fact (int n)
{
  if (n == 1)
  { return 1; }
  else
  { int total = fact (n-1) * n ; }
  return total;
}
int main ()
{ int no;
  pf ("Enter no:");
  sf ("%d", &no);
  fact(no);
  pf ("factorial = %d", total);
}
```

Q Recursive program for palindrome

```
boolean palindrom (int m, int n, String str)
  {   if(m >n)
      { return true; }
      elseif (str[m] == str [n])
      {
        palindrom (m++,n--, str);
      }
      else if ( str(m] ) =str[n]
      {
        return false;
      }
  }
```

```
int main ()
{ char ch [5] = "nitih";
    a=0
    b= strlen (ch)
    palindrom (a, b, ch);

}
```

Q. Recursion for fibbonssi series

0 1 1 2 3 5

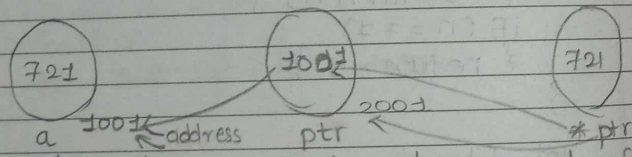
F= f(n-1)+f(n-2

```
int fibbonanci (int n)
{ if (n==0 || n==1)
  { if (n==0)
    { return 0; }
    if (n==1)
    { return 1 }
  }
  int fib_value = fibbonanci(n-1)+fibbonanci(n-
  return fib_value;
}
int main ()
{ int (no);
  pf ("Enter no")
  sf ("%d", &no);

  fibbonanci (no);

}
```

by Reehan sir

GoodLuck | Page No. | Date

- Pointer
- Pointer is nothing but it is kind of variable which store the address of another variable.
- Use to store a memory address of variable
- Pointer allow low level memory access
- Pointer = dynamic memory allocation
- Syntax -
  datatype * nm pointer;
- e.g. -
  int *ptr;
- where
  * - dereferencing



- How to use pointer
1) Declare the pointer
2) Initialize the pointer
3) Pointer dereferencing

```
int myhome;      → variable declaration
int *ptr;        → Pointer declaration
ptr = &myhome;   → pointer initialization
```

ptr is a kingdom and *ptr is king of kingdom

- Types of pointer -
1) integer pointer
2) Array of a pointer
3) Structure pointer
4) Function pointer
5) Double pointer
6) Null pointer
7) Void pointer
8) Wild pointer
9) Constant pointer
10) Pointer to constant
11) FAR pointer
12) Near pointer
13) Huge pointer
*** 14) Dangling pointer

① Integer Pointer -
A pointer which stores integer values is called int pointer.
e.g. int *ptr ← name of pointer

② Array pointer
```
Int num[3] = {10, 20, 30};
int *ptr
ptr = &num
num[0] = 10;
num[1] = 20;
```

int - specific value.
void - mixed value

```
#include <stdio.h>
int main()
{ int i=10;
  float j=20.3;
  char c='z';
  void
  int *arr[3];
  arr[0]=&i;
  arr[1]=&j;
  arr[2]=&c;

  pf("Int value of pointer : %d", *((int*)arr[0]));
  pf("float value : %f", *((float*) arr[1]));
  pf("characte : %c", *((char*) arr[2]));
}
```

for all type of pointer

Disadvantage of array pointer -
1) High memory consumption
2) Complexity
3) Memory leak

3) Structure pointer -
- A pointer which pointing structure type like structure, list, linked list, tree, graph
- In structure pointer we use keyword as struct
- syntax
  struct name_of_structure *ptr;

Structure pointer mainly used in complex

4) Function pointer -
- If a pointer is pointing to the function instead of pointing to specific variable then it is called as functional pointer
- It is mostly use for pointing the full code
- syntax -
  int (* name_of_functional_pointer) (parameter)
- e.g.
  int (*ptr) (int i, int j);
  int (*ptr) (int i, char j);

5) Double pointer -
- When a pointer stores address of another pointer called as double pointer
- Syntax -
  datatype **pointer_name;

Can we create multiple level pointer
Yes we can creat multiple level pointer
1) *ptr = single level pointer
2) **ptr - double level pointer
3) ***ptr - triple level pointer
4) ****ptr - fourth level pointer
5) ***** ptr.

6) Null pointer -
If a pointer is not pointing to any specific memory allocation or address that called null ptr

for declaring the null pointe to use NULL
Null pointer do not point to any memory location

datatype *ptr = NULL

or ptr = NULL
It is good sign of programmer if a coder is initialising pointer to null value

7) Void pointer
- Void pointer is a pointer of type void and it is also called as generic pointer
- Void pointer specifically held any type of address and it can be typecast one address into another address

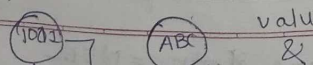
* What is mean by legacy?
The pointer is invented in first version of pointer.

Syntax -
void * ptr
Void pointers cannot be dereference directly

8) Wild pointer -
If you are declaring a pointer & we are not initializing it then it called wild pointer
e.g. ~~*ptr~~ int *ptr.
char * str

9) Constant pointer -
A memory address store inside pointer and it doesnot allowing to change the address is called constant pointer

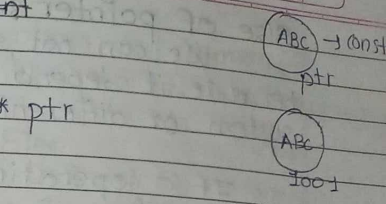
10)
If a stored address inside a ptr does not allowing to chang it then it called constant pointer



as per our program the memory for our program is allocated in Heap.

10) ~~Pointer to constant~~
int *const ptr.

~~Cont~~ const int * ptr



11) FAR pointer -
A pointer which holds 32 bit memory. FAR can access memory outsid the current segment

12) NEAR pointer
A pointer which have 16 bit memory allocation.

13) Huge pointer
Containt current segment address & offset address

14) Dangling pointer -
If a pointer is having address of another varia that variable is deleted from memory but still that pointer is pointing that memory address is called dangling pointer.

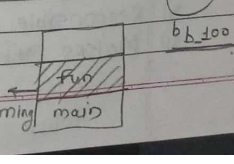
A pointer is pointing to variable that is already deleted.

```
#include<stdio.h>
int *fun(void)
{ int b=600
  return &b;
}
int main()
{ int a=500
  int *ptr=NULL;
  pf("%p",&a);
```

address

```
  ptr=fun()
  pf("%d",*ptr);
}
```

o/p - 600



If Heap when function return to main then the function will deleted automatically

GoodLuck | Page No. | Date

Size of pointer :-

We can not say a size of pointer because it depend on different operating system or different CPU architecture

If a operating system is 64 bit then the size of pointer is 8 byte

If a system is 32 bit then the size of pointer is 4 byte

If we dont know the size of operating system & you want to find size of pointer then use sizeof()

e.g. sizeof(ptr)

Advantages of pointer-

1) Pointer are use to dynamic memory allocation and deallocation
2) Arrays or structure can be access by pointer
3) Pointer are useful for access direct memory address
4) Pointer are used for complex data structure such as linked list
5) Pointer reduce the length of program and execution time as well

Disadvantage of pointer

1) Memory curreption can be occured
2) Pointers are very complex to understand
3) Responsible for memory leak
4) Makes code slow

by Pankaj sir

18/02/24

GoodLuck | Page No. | Date

File handling -

RAM	Hard disk
Volatile memory	Non-volatile memory

Files

Text file	Binary file
.txt, .java, .doc, .c, .py	.class, .exc, .apk

- File pointer -

FILE *pointername

Use to refer the perticular file

- Operation on file:
  1) Create a file
  2) Open a file
  3) close a file
  4) Write into a file
  5) Read from a file.

- Modes
  1) Read - 'r'
  2) Write - 'w'
  3) Read Binary - 'rb'
  4) Write Binary - 'wb'
  5) Append - 'a'

- Open a file.

```
FILE *fptr;
fptr = fopen("filename.txt", "mode");
```

e.g.

```
fptr = fopen("Hello.txt", "r");
```

If the file is does not exist then fopen return NULL to fptr (only in 'r' mode),

- Read from file - & write into file

fscanf - read from file.

fprintf - print / write into file

Syntax -
fscanf: (file-pointer, format-specifiers, &var_name)
fprintf: (file-pointer, access_specifier, varname)

eg-
fscanf (fptr, "%c, &ch)
fprintf (fptr, "%c", ch)

- closing the file
  fclose (fptr)

- WAP to write single integer no inside a file

```
#include <stdio.h>
int main()
{
  FILE *fptr;
  fptr = fopen("sample.txt", "w");
  int no = 5;
  fprintf(fptr, "%d", no);

  fclose(fptr);
}
```

- fputc, fgetc -

fputc (var_nm, file_pointer) fprintf
int x = fgetc (fptr) fscanf

- WAP to write a single character inside file

```
FILE *fptr
fptr = fopen ("char.txt", "w");
  char ch = 'a';
  fprintf (fptr, "%c", ch);
// fputc (ch, fptr);
fclose (fptr);
```

WAP to check whether the Hello.txt file is exist or not

```
FILE *fptr
fptr = fopen ("A.txt", "r");

if (fptr == NULL)
{
   pf ("file not exist");
}
else
{ pf ("file exist"); }
fclose (fptr);
```

WAP to read the integer value from the file

```
FILE *fptr;
fptr = fopen ("A.txt", "r");
int i;
fscanf (fptr, "%d", &i); printf ("%d", i);
fclose (fptr);
// or  fge  int x = fgetc (fptr)
        printf ("%d", x);
```

WAP to write a character in file hello.txt. Inside hello.txt the data is present.

```
FILE *fptr
fptr = fopen ("Hello.txt", "a");
char ch = 'a';
fprintf (fptr, "%c", ch);

fclose (fptr);
```

or fputc (ch, fptr);

WAP to read everything from file

```
FILE *fptr;
fptr = foper ("Hello.txt", "r");
while (ch != EOF){
    char ch = fget (fptr)
    printf ("%c",ch);
}
```

WAP to calculate whether the given no is prime no or not

```
int main()
{ int no;
  pf ("Ente no");
  sf ("%d", &no);
   i=1
  do
  {
   if(no % i. == 0)
   count ++;
  i++;
  } while (i<=no/2);

  if (count==1 .)
  { pf ("prime" ); }
  else
  { pf ("not prime"); }
```

```
scanf ("%d", &n);
for (i=2; i<= n/2; i++);
{  if (n %i == 0)
    { pf ("Not prime");
       break;
    }
    else
    { flag += 1; }
}
 if(Flgg == (n/2)-1)
{ pf ("prime"); }
```

OR

```
int main ()
{ int n, flag=0, i;
  pf ("Enter no");
  sf ("%d", &n);
  if (n==0 || n==1)
  { flg = 1 }

for (i=0; i<= n/2; ++i)
{  if (n%i == 0)
     { flag = 1
        break
     }
}
if (flag == 0)
{  pf("prime"); }
else
```

find the sum of digit.

```
no = 12345
sum = 0
rem = 0

while (no > 0)
{  rem = no % 10;
   sum += rem
   no = no / 10;
}
pf (sum);
```

WAP to reverse the no.

```
while (no > 0)
{  rem = no % 10;
   rev = rem * 1000
   no = no / 10
}
```

WAP for armstrong no.

by Reehan Sir
18/12/24
GoodLuck | Page No.
Date

~~find out~~

Structure

→ Structure is type of datatype which is user defined & it allows to combine differen datatype.

We can not use different datatype in array so we use structure to save it.

syntax -

```
struct structure_nm
{ - - -
  - - -
};
```

e.g -

```
struct student          OR    struct student
{ int i;                      { int i;
  int j;                        int j;
};                            } x;
struct student x = {10, 20}
```

x.i = 10
x.j = 20

Memory allocation for structure -

Structure allows differen different memory allocation for different differen data type.

```
struct Employee
{ int id;
  char nm[20];
  float x;
} v;
```

id → int = 2byte
nm → char 1 byte
x → float 4 byte

memory size of v = 7

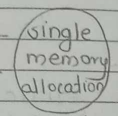
2+1+4 = 7

- **Union:-**

Union is kind of user datatype where single allocation of memory according to variable highest memory will be assigned to whose memory is more.

Use to reduce the cost of program & reduce to memory ~~wastage~~ allocation

```
Union Employee
{ int id;
  char nm[20];
  float x;
};
```



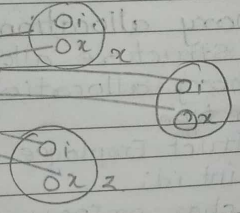
memory for V = 4 byte

float

- **Types of structure -**

1) Structure pointer
2) Nested structure
3) Seperate structure

```
Struct car
{ int id;
  float x;
}x, y, z;
```



Structure pointer:

If is a pointer is pointing to a structure and using arrow (->) operator then it called as structure pointer

```
#include <stdio.h>
struct Employee
{ int x, y
};
int main()
{ struct employee ed = {10, 20};
  struct employee *ptr;
  ptr = &ed;
  printf("%d %d", ptr->x, ptr->y);
}
```

Advanced structure:

1) Nested structure
2) Nesting seperate structure if we are going for highest implementation of structure.

- **Nested structure**

If a structure allows allows insert the another structure inside that initialize structure then it called nesting or called as nested structure

ex -

```
struct employee <- Nested
{ int id;
  char nm[20];

  struct Date <- nesting
  { int dd;
    int mm;
    int yy; }dmy;
} emp1;
```

GoodLuck | Page No. | Date

```
empt.dmy.dd = 3;
empt.dmy.mm = 2;
empt.dmy.yyyy = 2024;
```

→ This is a passing value method to respected variable.

```
empt.id = 101;
empt.nm = {"chinya"};
strcpy (empt.nm, "patil");
```

- **Seperate nesting -**
  When we declare two structure seperately (in order of parent & child) then this called seperat nesting or seperate nested.

```
#include <stdio.h>
struct child
{ int x;
  char nm[10];
};
struct parent
{ int a;
  struct child ct;
}
int main()
{ struct parent pt = {10, 100, "abc"}
  printf("var of parent a = %d", pt.a); //10
  pf("var of child x = %d", pt.ct.x); //100
  pf("var of child nm = %s", pt.ct.nm); //abc
}
```

GoodLuck | Page No. | Date

- **limitations of struct.**
1) High memory consumption
2) Data simply visible for everyone because it does not hold data hiding
3) We can not declare a function inside struc
4) Since struct can not have static memory inside in body
5) struct can not have constructor inside struct

- **Advantage -**

- **Union -**
1) Union is user defined datatype & it collect different variable into same memory location
2) We can define it with many member but we can access # only one member
3) Union can be very heavy if you are using memory mapped register

Need of Union
1) Used to save memory
less memory, better to understant with execution
When we want to assign new value to field then exsting value will be assign replaced by new value

- C: union share mutualy same memory Union are mostly used embedded programming

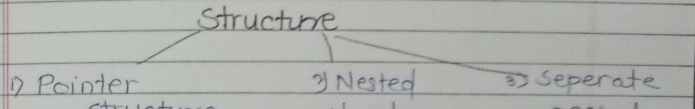
Difference bet° Union & struct

struct	Union
struct allocate all possible seperate memory to all variable	Union allocate space to largest field of variable & all field are stored in same memory location
The first variable is always store in start of struct	

```
#include <stdio.h>
Union t2
{ int x,y; };

Struct t1
{ int x,y; };

int main()
{
  Struct t1 s = {1,2};
  Union t2 u = {0,0};
  u.x = 3;
  pf ("value of x = %d", u.x); // value of x:
  pf ("y = %d", u.y);          // 3

  u.y = 4;
  pf ("x from union : " u.x);  // 4
}
```

Conclusion :-
All unions are seperate from union

Structure = seperate allocation of memory
Union = largest allocation of memory



((Adding struct + new struct = Nesting)) → nested struct