Assignment - 3

Page No.
Date: 6/9/24

1) What are different standardizations of C Programming

→ - ANSI C, ISO C and Standard C are successive standards for C programming language published by American National Standards Institute (ANSI) and ISO/IEC JTC 1/SC 22/WG 14 of the International org" for standardizat" (ISO) & International Electrotechnical Commission (IEC)
- Best supported versions → C89 or C90

2) Standardizing C
1983 - ANSI formed committee
1985 - 1st std draft (C85)
1986 - C86
1988 - C published (C88)

2) C89 → ANSI C

3) C90 - Ratified by ISO/IEC

4) C95 - 1995 (Improved multi-byte & wide character support, Add" of digraphs, specifications of std macros, 2 technical corrigenda published)

5) C99 - Mar 2000, ANSI adopted ~~ISO/IEC~~
- New core language feature
- New library headers, improved compatibility with C.

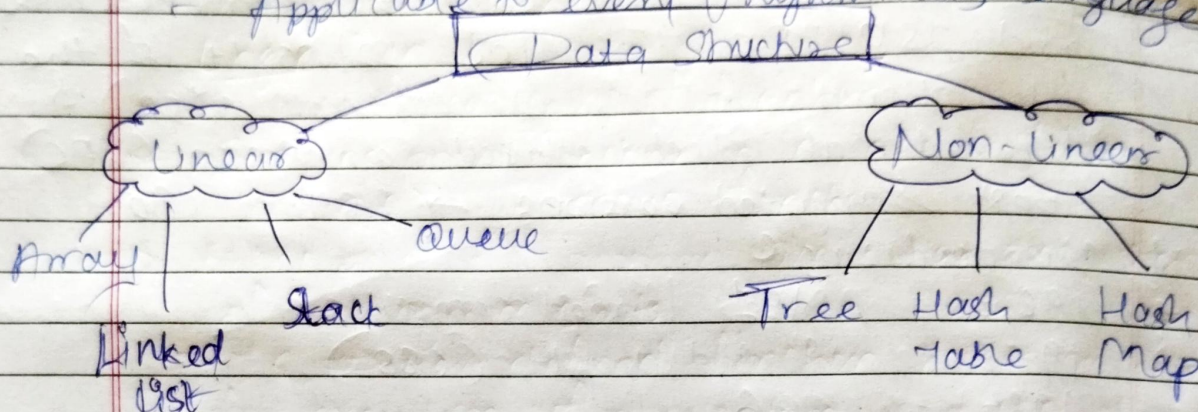
6) C11 (2011) - Improved unicode support, type-generic expressions using new keywords, etc.

7) C17 - 2018 - current std for C Programming
Addresses defects in C11 w/o any changes

8) C23 - Informal name for next major C language std revision

2] What is meant by Data Structures?

→ - Defined as a way of storing & representing a data in a particular format.
- Concept of Data Structure - 'Language Independent'
- Applicable to every Programming Language

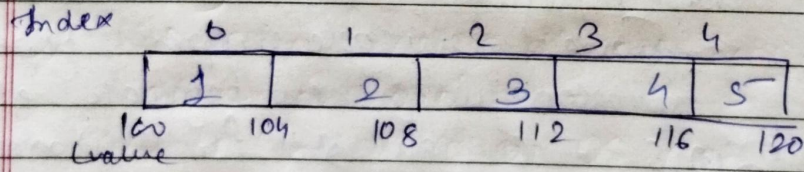


3] What is Array? explain in detail.

→ - Considered as a linear Data Structure
- Derived Data type in C, C++ & Java.
- Holds - 'Multiple Homogeneous variables in Index form'.

Eg: int Arr[5] = {1, 2, 3, 4, 5}

Syntax: Datatype ArrayName [length of array] = {Data elements}



1) Memory of elements - 'sequential way'
2) Separate name & memory for each element.
3) memory allocated in consecutive manner
4) Access all element - using single names
5) Array index starts - zero (0)
- '' — last - (n-1) ... (n = length)

Page No.
Date : / /

6) Create array of Primitive Data type :
EXCEPT : Boolean & void

7) Array is :
1) One-dimensional [] - subscript operator
2) Two dimensional [][]
3) Multi-dimensional [][][] ... []

4) Differenciate Constant and variable
→

Constant	Variable
1) A variable that can't be altered, once defined	1) A variable is a name associated with some memory location.
2) Used to hold fixed value that can be retrieved not changed	2) Holds value that can be changed
3) Generally Stored in text segment as are 'readonly'	2) Stored in data segment, heap or stack, depends.
4) Keyword - Const	4) Standard variable definition syntax
5) E.g. const int pi = 3.14;	5) E.g. int a = 10; (variable)

5) Difference between Local & Global variable.

Local Variable	Global Variable
1) United to block of code	1) Accessible throughout the program.
2) Typically within functions or specific blocks	2) Outside of any funcn or block
3) Accessible only within the block where they are declared	3) Accessible from any part of the program
4) Created when block is entered	4) Retain their value

Page No.
Date : / /

Q) Explain Concept of function, explain about starting point function in C, C++ & Java.

→ - Function : Named Block which contains the set of instructions, which are related.
- When we define function we write 'Business Logic' in it.
- Achieve reusability of function.

* Starting point function in C, C++ & Java:
main() - main function

1) Entry point of a program where the execution of a program starts.
2) Is a user-defined function that is mandatory for execution of a program because when a C program is executed, OS starts executing the statements in the main().

Syntax :

```
returntype main()
{
    .....
    .....
    return 0;
}
```

3) The main function always returns an integer value or void.
4) main() called by OS not the user.
5) Ways of writing :

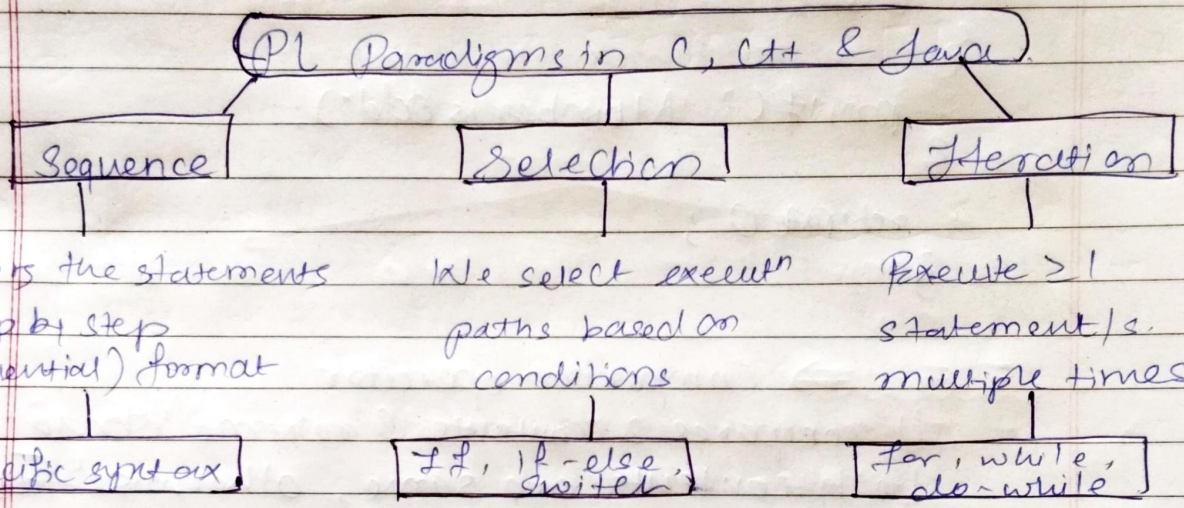
1) int main() {}	2) int main(void) {}
3) main() {}	4) void main() {}
5) void main(void) {}	

6) Types :
- main() with no arguements & void return type
- main() with no arguements & int return type
- —— " —— Command Line Arguements.

Page No.
Date : / /

Q) Explain about PL Paradigms in C, C++, Java in detail.

→ Paradigm - High level way of conceptualizing & structuring a computer program.



1) Sequence:

Sequential execution of multiple statements.

Eg:
```
{ int No1 = 1;          } (1)
  int No2 = 2;

  printf("Values of No1 & No2 = \n"); } (2)
  scanf("%d", &No1);    } (3)
  return 0;             } (4)
}
```

Here steps (1), (2), (3), (4) will get executed in a sequential manner.

2) Selection:

- Select desired option, if there are multiple options based on requirement.

Page No.
Date : / /

If-else-if

```
if(remainder == 0)
{
    printf("Number is even");
}
Else:
{
    printf("Number is Odd");
}
return 0;
}
```

== ⇒ Comparison operator
- It compares 2 contents & returns 'true' if both contents are same, otherwise 'false'
- It compares 2 contents & returns (true) if both contents are same, otherwise (false)
  Eg. 11 == 11 → true
  11 == 10 → false.

Switch case :
- Code inside the appropriate case gets executed directly
- Execution - faster than if-else if.
- Last case = default case, gets automatically executed if no matching case
- 'Break Statement' - Break execution & come out of switch case

3] Iteration:
Execute ≥1 statements multiple times

Short-hand operators

Increment: Pre (++i) post (i++)
Decrement: Pre (--i) Post (i--)

Page No.
Date : / /

* Suppose No. is a variable, which is of type integer & initialized with value 100
  int No = 100;
  No ++ ; // it is just a increment (nor pre / nor post)
* Compiler will convert short hand operators to normal statements.

1) **For loop:**
- Used when we know no. of iteration that we want to execute

Eg.
```
int main()
{
  int i = 0          // Loop counter
  for (i = 1 ; i <= 10 ; i++)
  {
    printf ("Jod. jay ganesha ...\n", i);
  }
  return 0;
}
```

① Initialize loop counter | ② Conditions
③ Increment / decrement of loop counter | ④ Loop Body

2) **while Loop:**
- When we know condition, till which we want to execute the code.

Code:
```
{
  int i = 1 ;   // loop counter
  while (i <= 5)
  { printf ("Jod. jay ganesha ...\n", i);
    i++;
  return 0;
}
```

- The code will get executed till the value of i is equal to 5.

3] Do-while loop :
- It gets executed atleast once irrespective of the conditions
- It is similar to while loop.

E.g. Code :

```
{
int i = 1;
do
{
printf("dod. jay ganesha ... \n", i);
i++;
} while (i<=5)
return 0;
}
```

* used for Atleast Single Iterative traversal

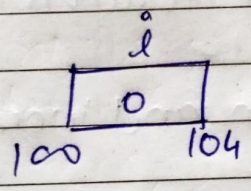
8] Program to accept a input for user & find whether the no. is even or odd ; draw diagramatic layout of same program.

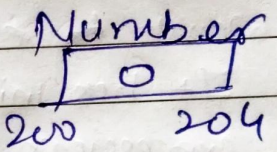
→

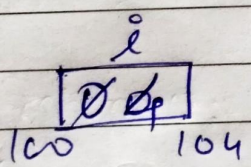
```
#include<stdio.h>
int main()
{
int i = 0;
int Number = 0;
printf("Enter number : \n");
Scanf("%d", &i);

Number = i % 2
```

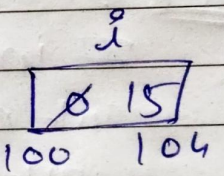
```
if (Number == 0)
{
    printf ("Number is Even");
}
else
{
    printf ("Number is Odd");
}
return 0;
}
```

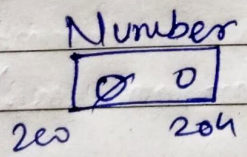
Diagrammatic Layout :

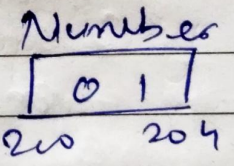












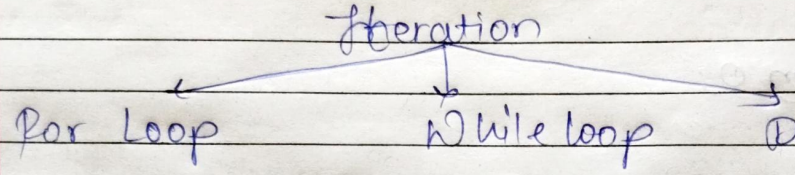
0 == 0 ⇒ true
Number is Even

1 == 0 ⇒ False
Number is odd.

Page No.
Date : / /

10] Notes on various loops, with syntax & dry run.

→ There are 3 PL paradigms viz :
1] Sequence - Sequential execution of multiple statements
2] Selection - Select desired output from multiple options.
3] Iteration - Execute 1 or more statements multiple times



1] For Loop :
- used when we know no. of iteration that we want to execute
- It has initialization, condition and increment expressions in a single line.

Code :

```
#include <stdio.h>
int main () {
  int i = 0
  for (i=1 ; i<=100 ; i++)
  {
    printf ("Hod. jay Shriram ...\n", i);
  }
    return 0;
  }
```

2] While Loop :
- When we know condition till which we want to execute the code.

```
#include <stdio.h>
int main()
{
    int i = 1 ;
    while (i <=5)
{
  printf "%d. I am happy.....\n", i);
   i++ }
  return 0;
  }
```

Step 1: i = 1

2 - 1 <= 5 → true

4 - 1. I am happy....

3 - i ++

i
1

} same for Iteration ②, ③, ④, ⑤

Step 6: ~~Iteration~~ ⑥ 6 <= 5 → False, break.

3. ① Do-while

Almost same as while loop but it gets executed at least once irrespective of conditions.

```
{
   int i = 1
   do
{ printf ("%d. I am happy...\n", i);
   i++;
  } while (i <= 5)
return 0;
}
```

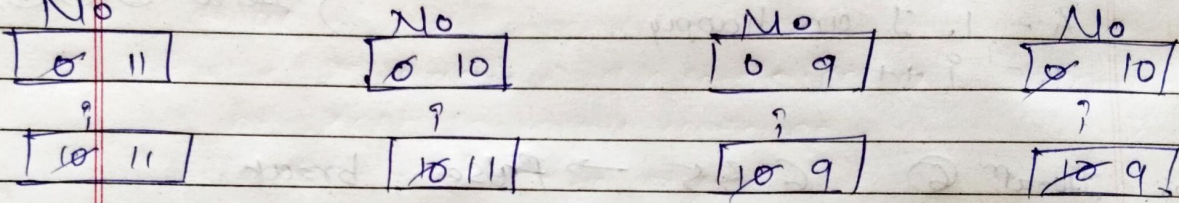
Page No.
Date : / /

Q] Note on : Short Hand operators.

→ In C, C++ & Java there are 2 short-hand operators, which are used to increment & decrement value by 1 respectively.

Short-Hand operators

Pre-increment	Post increment	Pre-decrement	Post decrement
No = ++i;	No = i++;	No = --i;	No = i--;
1st increment then assign	1st assign then increment	1st decrement then assign	1st assign then decrement



* Suppose No. is a variable, type integer, initialized with value 50

```
INT No. = 50;
No++;  .. // it is just a increment (nor pre nor post)
```

Above statement is converted internally as:

```
No = No + 1;
```

For No--:

```
No = No - 1;
```

Note : Compiler will convert short-hand operators to normal statements.