UNIT-2

DATE: __/__/____
PAGE: _____

→ Loops ⟶ for loop, while loop, dowhile loop

★ do-while loop will Execute Atleast once when Cond^n is False but other 2 loops wills't Run when The Cond^n is False.

Note ⟶ For loop is also Known as determinate loop or definte loop.

GoTo Statement ⟶ "goto" Statement is also called as Jump Statement in C. It is used To transfer Program Control to a predefined label. goto statements is Preferable when we need to Break mullifle loops using a Single Statement at The Sme Time.

→ The Use of "goto" statement is been avoided because it Causes Confusion for The fellow Programmers in Undelstanding The Code.

→ If one for loop is Present inside another for loop and we Want to Exit from both The loops, Then, instead of using break statements, we Can use "goto" statement.

⊙Eg:-
```
for (int i=0 ; i<10; i++){
        for (int j=0 ; j<=10 ; j++){
                printf ("\nEnter an Number (0 To Exit loops) ::: ");
                Scanf ("%d", &num);  ——(1)
                if (num==0){ goto end ; }  ——(2)
        }
}
end : printf ("\nYou have Exited The 2 massive for loops");  ——(3)
```

∴ OUTPUT ⟶
```
Enter an Number (0 To Exit loops) ::: 3
Enter an Number (0 To Exit loops) ::: 29
Enter an Number (0 To Exit loops) ::: 0
You have Exited The 2 massive for loops
```

Explintion ⟶ It will be Very diffinult To Execute The Statement ② without goto Statement, but, goto helped us in This problem. Statement ① will Input Number from User, If, Input Number is 0, Then, ② will Execute and goto will move The Compiler To ③ and we will Exit The loop Very Easily. Then, ③ will Execute and Then Program Ends.

— JAI LOGIC

Good Write

Imp.

DATE: ___/___/____
PAGE: _____

Q → Write diff. b/w break, Continue and goto ?

Break	Continue	goto
① Leads To Immediate exit of Enclosing loop	① begins next Iteration of the while, enclosing for or do loop.	① Jump statement That Takes Control from anywhere to another.
② Uses to Stop further Execution of loops.	② Used To Execute further loop.	② Used as an unconditional Jump Statement.
③ Can apply on Both loops and switch.	③ Can apply on loops, not on switch.	③ Can apply in switch and loops.
④ SYNTAX ::: break;	④ SYNTAX ::: continue;	④ SYNTAX ::: goto jump;
⑤ doesn't require an label.	⑤ doesn't require an label.	⑤ Require an label.
⑥ Can be used with 'if' only inside The loop.	⑥ Can be used with 'if' only inside The loop.	⑥ Can be used with 'if' anytime
⑦ Control is Transferred Outside The loop.	⑦ Control is in loops	⑦ Control is out from loop.

→ Functions in C- language

→ C- Enables The Programmer to break up an Program into Segments Commonly known functions each of which can be written more or less Independently than others.

→ The moment The Compiler Encounters a func. Call instead Of Executing the next stat. in Calling func, The Control Jumps To The Statement that are part of called func.

→ After The called func is Executed, The Control is Returned back To The Calling function.

Q Eg:- int fib (int n) { // n is Index Value of fib Series...

```
    if (n==1){
        return 1;
    }
    return fib(n-1) + fib(n-2);
}
```

1. User defined f^n → These are Created by Programmers are called user defined functions. It Reduces Complexity of Big Programs and Optimises the Code.

2. Library f^n → These are defined in C-Header file such as printf, Scanf.

Good Write

DATE: ___/___/____
PAGE: _____

1. Need of Functions in C-Language

→ It is Good Approach of dividing a Program into seperate well-defined functions.
→ Understanding, Coding, testing multiple seperate func is for Easier Than doing with one big f^n.
→ Maintainance of Program will Get Easier (All lib. in C Contain a set of f^n).
→ Programmer can also write Thier f^n and can use at diff. Points in Main() f^n.
→ A f^n can Call any other f^n and any N.o. of Times as well.

2. Facts about Functions

→ A f^n 'b' that uses another f^n 'a', Then, b is Called as 'Calling function' whereas a is Called as 'Called function'.
→ Inputs Taken by f^n are known as Parameters / Arguments.
→ The Calling f^n may / mayn't Pass The Parameters to The Called func.
→ f^n declaration ≡ f^n + List of Arguments + data it Returns.
→ Semi Colon → f^n declaration & otherwise → f^n definition

⭐ → Parameters Passing in f^n → Call by Value (Copy of Actual Parameter is Passed)
→ Call by Reference (Address of Act. Para. is Passed)

⇒ Storage Classes

→ Used To Tell The Scope / Lifetime of a Variable of f^n.
→ 4 Types are auto, static, extern and register.
→ It defines The Scope and lifetimes of Variables / f^n declared in C-Program.

1. Need of Storage Class in C-language

→ The Storage classes of a f^n of a Variable determines The part of memory where the storage space will be allocated for The Variable of f^n.
→ It specifies how long The storage allocation will Continue.
→ It also Specifies That in which part of program. The Variable is Visible & Accessible.
→ Specifies whether the Variable / f^n has Internal, External of No Linkage.

⭐ → Auto is The default Storage Class for all local Variables and Static for all Global Variables.

Good Write

DATE: ___/___/_____
PAGE: _____

Storage Class	Storage Place	Default Value	Scope	Lifetime
Auto	RAM	Garbage Value	Local	Within the f^n in which is defined
Static	RAM	0	Global + Local	Till End of main() f^n, retains value in multiple f^n calls
Extern	RAM	0	Global	Till End of main() f^n and may be declared Anywhere in program
Register	Registers	Garbage Value	Local	Within function

1. Automatic Storage Class

- Automatic Variables are Allocated memory Automatically at any time.
- Visibility is limited To the Block in which defined.
- Scope also limited To the Block in which defined.
- Memory defined in Automatic Variables gets freed upon exiting from the Block.
- Auto Variables can be Accessed in the Block / f^n they have been declared.
- In a case, They are only Accessible outside their scope with pointers only.

2. Static Storage Class

- Static Variable can only hold their value b/w Multiple f^n calls.
- A same Static Variable can be declared many times but can be assigned only one time.
- Default Initial Value of Static is 0 or Null.
- Diff. b/w an auto and Static Variable is that Static Variable when defined within an f^n is not reinitialized when an f^n is called again and again.

3. External Storage Class

- Used to Tell the Compiler that the Variable defined as extern is declared with external linkage
- The Variable declared as extern is not allocated any memory. It is only declaration and Intended to specify that Variable is declared elsewhere in Program.
- We can't Initialize external Variable within any block / method.
- We can only Initialize External Variable ~~within any~~ Globally.
- It can be declared many times but only can be Initialized once.

P. T.O

Good Write

DATE: ___/___/_____
PAGE: _____

4. Register Storage Class

→ This Storage Class can declare the register Variable of same functionality as of auto Variable.
→ Variables defined as Register are allocated memory in the CPU Registers depending upon the size of memory remaining in the CPU.
→ The Access time of Register Variable is Faster than Automatic Variable.
→ Few Variables which are Accessed very freq. in the program are declared with the Register Keywords which improves Running time of Program.
Drawbacks → Compiler Error / Illegal Execution when we store Input in Register Variable

→ Arrays (Subscripted Variable)

→ It is a Collection of similar datatypes stored in Contigious memory locations.
→ SYNTAX ::: datatype arrayname [size] ;
→ Array is used in Weather Forecasting, Employees of a Company.
→ Arrays are Code optimized, Easy for Traversing, Easy for Sorting and Random Access.

Note → Doesn't allow declaring an Array whose N.O. of Elements are Not Known at Compile Time.
Length of an Array = Endindex - Startindex + 1

Q → Pass Array as Argument to fn in C- language ?

→
```
# include < stdio.h >

Void array (int arr[] , int index) {
    for (int i=0 ; i < 5; i++) {
        int m = arr [i];
        ~~if ( m == i if (i == index) {~~
        if (i == index) {
            printf (" Value at Index %d is %d ", index, m);
            break;
        }
    }
}
int main () {
    int arr [5] = {1,2,3,4,5};
    array ( arr , 2)
}
```

Output → Value at Index 2 is 3

2D-Array → datatype arrname [rows] [columns]

DATE: ___/___/_____
PAGE: _____

⇒ Strings (%s)

→ String defined 1D-Array of characters which is terminated as Null (\0).

→ We can declare an string by char Array or by String Literal.

→ Char Array ⟶ char arr [10];

String Literal ⟶ char name [5] = "Yash"; ⟶ #include <string.h>

1. strcpy (S1, S2); ⟶ Copy the String S2 in S1
2. strcat (S1, S2); ⟶ Concatinate the String S2 at End of S1
3. strlen (S1); ⟶ Returns the Length of String S1
4. strcmp (S1, S2); ⟶ Returns 0 (S1 = S2), <0 (S1 < S2), >0 (S1 > S2)
5. strchr (S1, ch); ⟶ Returns a pointer to the First Occurance of char ch.
6. strstr (S1, S2); ⟶ Returns a Pointer to First occurance of S2 on S1.

(For above 6 String.h", "#include <string.h>" header file is must")

Note → scanf ("%s", &name); ⟶ This will help to Input String and Store to name

⭐ → To make a Code for Space Seperable Strings, The minor changes are req. in scanf() as:-

"scanf ("%[^\n]s", &st);"

IMP. → In C-language, An Array points to an location in Memory and the 1st Element of Array also points to the same memory location. Thus, the 1st Element of Array is 0 Element Away from the Memory location of Array, due to this, the 1st Element Got index = 0. Similarly, 2nd Element of Array is 1 Element away from the Memory location of Array, due to this, the 2nd Element Got index = 1. This is the Reason behind 0-Based Indexing in Programming language.

— JAI LOGIC