C Language Notes DEMO

Handwritten Notes

Date ______
Page ______

* Control statements (Decision Control structure)-

i) If statement-

```
if(condition)
{
        statement;
}
```

An 'if' statement consist of a boolean expression followed by one or more statements.

If the boolean expression evaluates to true then the block of code inside the 'if' statement will be executed.

Eg:-
```
#include <stdio.h>
int main()
{  int a = 10, b = 20;
   if (a < b)
    { printf("a is less than b");
    }
   return 0;
}
```

ii) If ---- else statement-

An 'if' statement can be followed by an optional 'else' statement, which executes when the boolean expression is false.

If the boolean expression evaluates to true, then 'if' block of code will be executed; otherwise 'else' block of code will be executed.

Date
Page

Syntax

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

Eg:- 1) To check a number is even or odd-

```
# include <stdio.h>
void main()
{   int n;
    printf(" Enter a number");
    scanf(" %d", &n);
    if (n%2 = = 0)
        printf(" Even number");
    else
        printf(" Odd number");
}
```

2) Consonent or vowel program-

```
# include <stdio.h>
int main()
{ char ch;
  printf(" Enter any alphabet \n");
  scanf(" %c", & ch);
  if (ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
      printf(" Vowel");
  else
      printf(" Consonent");
```

Date
Page

* Array -
Array is a collection of similar data items or elements. Or we can say that array is a collection of homogeneous data ~~elect~~ element stored continuously under a single name.

* Each data item of an array is called an element and each element is unique and located in separated memory location.

* Array of character is known as string.

* Each location of an element in an array has a numerical index no., known as subscript. which is used to identify the element.

* Index is always starts with zero.

* An array can be a single dimensional or multi-dimensional.

* One dimensional array is known as vector and two dimensional arrays are known as matrix.

* Declaration of an array -
Syntax -
data type array name [size];

```
Eg:- int arr[100];
     int sub[100];
int a[5] = {10, 20, 30, 100, 5}
```

* Array subscript (index) always start from zero which is known as lower bound and upper value is known as upper bound.

$$\underbrace{\text{int}}_{\text{type}}\ \underbrace{\text{ar}}_{\text{name}}[\overset{\text{Size}}{5}] = \{\underbrace{\overset{\text{Index } 0}{20}}_{\text{lower bound}}, \overset{1}{60}, \overset{2}{90}, \overset{3}{100}, \underbrace{\overset{4}{120}}_{\text{upper bound}}\}$$

→ Input values into an array and display them -

```
# include <stdio.h>
int main()
{
  int arr[5];
   int i;
  for(i=0; i<5; i++)
   {
     printf("Enter a value for arr[%d]\n", i);
     scanf("%d", & arr[i]);
   }
     printf(" The array elements are\n");
     for (i=0; i<5; i++)
       {
         print("%d\t", arr[i]);
       }
  }
```

→ "main()" function is a user defined function.

* Function -
Function is a group of statement that together perform a task. Every 'C' program has at least one function, which is "main()".

* There are two types of function -
i) Library function - (Predefined function).
ii) User defined function

i) Library function -
Those function which come along with the compiler and are present in the disk. These function can't be modified, it can only read and can be used.

* Every ~~program~~ library function has a header file.

* There are a total of 15 header file in 'C'.

* Printf(), scanf(), clrscr(), strcpy() etc are example of library function.

ii) User defined function -
The user defined functions defined by the user according to its requirment.
syntax -
return type name of fn. (data type)

Date
Page

* Pointer -
A pointer is a special variable that is used to store the address of some other variable.

* A pointer can be used to store the address of a single variable, array, structure, union or even a pointer.

* A pointer is a derived data type in C.

* Pointers allow C to support dynamic memory management

* Declaration of pointer variables -
syntax - data type ~~to~~ name * pointer name

(Here '*'(asterisk) before pointer indicate the compiler that variable declared as a pointer)

```
Eg:-  int *P1;        // pointer to integer type
      ~~int~~ float *P2;  // pointer to float type
      char * P3;      // pointer to character type
```

(When pointer declared, it contains garbage value i.e it may point any value in the memory)

* int *P1; (Here the type int refers to the data type of the variable which is pointed by P1 not the type of the value of the pointer.

* Pointer ~~declation~~ declaration style -

```
1) int* P;
2) int  *P;
3) int * P;
```

Date ______
Page ______

* There are two operators are used in the pointer -

i) Address operator (&) -
Address operator is used to access the address of a variable. It can be used only with a simple variable or an array element not with a constant value.

* Indirection operator(*) -
It gives the value stored at a particular address.

Eg:-
```
main ()
{
  ~~int a = 5;~~
  ~~int~~
```

* Initialization of pointer variable -

Eg:-
```
→ int a;
  int *P;     //declaration
  P = &a      // initialization
```

```
→ or, int x, *P = &x ;  /* declaration with
                            initialization */
```

```
→ int *P = &x, x;
```
(Not valid, because target variable x should be declared first.)

```
→ float a;
  int *P;
  P = &a;
```
(Not valid, because we can not assign the address of a flot variable to an integer pointer.)

Date_______
Page_______

Structure -

# Array allow you to define type of variables that can hold several data items of the same kind but structure is another user defined ~~at~~ data type available in C programming, which allows you to combine data items of different kind.

# It is the collection of dissimilar data types or hetrogenous data types grouped together.

# Each individual data item within the structure is referred to as member.

# A structure definition is specified by using the keyword 'struct'.

* Structure declaration -

```
struct student   /*Definition          (or)   struct student
                   of structure*/             {int roll;
 { int roll;                                    float marks;
   float marks;    /* members */                 char name[10];
   char name[10];                             }S1, S2;
 };
void main()
{
  struct student S1;  /* Variable of the
                          structure student */
}
```

* Members of structure are allowed space in the memory only when the objects are declared (Eg- S1, S2)

Date ______
Page ______

* Static and Dynamic memory allocation -

i) Static memory allocation -
The process of allocating memory during compile time is known as static memory allocation.

ii) Dynamic memory allocation -
The process of allocating memory during run time ~~to~~ or execution time as required is known as dynamic memory allocation.

* Difference b/w static and dynamic memory allocation -

static	Dynamic
i) Memory is allocated during compile time.	i) Memory is allocated during run time.
ii) Amount of memor allocated is fixed, can't change.	ii) Amount of memory allocated can be changed
iii) Wastage of memory may occure.	iii) No wastage of memory
iv) Memory is allocated from stack segment.	iv) Memory is allocated from heap segment.