"Storage Class"

To define a variable we need to mention not only its data type but also its storage class. Storage classes will tells the compiler, where the variable is stored and what will be its default value. In C, different types of storage classes and different types of variable declaration are :

1) Auto variable :
The auto keyboard is applied to all local variables automatically. It is the default storage class that is why it is known as automatic variable. However you can use auto keyword to declare automatic variable.

2) Register variable :
The register storage class is used to define local variables to be stored in a register instead of RAM.

3) Static Variable :
The static storage class instructs the compiler to keep a local variable in existence during the lifetime of the program. Therefore, making local variables static allows to maintain their values between funct^n calls. It retains value between multiple functions call.

4) External variable : It is also known as global variable and it is visible in ALL the functions. Global variable is declared outside the function or scope block. value change in any function will reflect in other functions. Extern keyword is use to declare external variable.

Scanned with CamScanner

150. WA

```
#include <stdio.h>
int main ()
{
    void demo ()
    register int i=1;
    while (i<=10)
    {
        demo();
        i++;
    }
}
void demo()
{
    static int x;
    printf ("\n x: %d", x);
    x+=2;
}
```

24/01/2022

"Structure and Union in C"

(*) Structure :

- introduction

If we want to write a program to store student information, which will have student's roll no., name, branch, address, etc, which included string values, integer values, etc. how I can ~~use~~ use arrays for this problem, I will require something which can hold data of different types together. so we can use the concept structure in C lang.

- Definition : Structure is a user-defined datatype, which allows you to combine data items of different kinds. To define a structure use ~~struct~~ struct keyword.

Scanned with CamScanner

Defining a Structure

Syntax:

```
struct structure_name
{
datatype member 1;
datatype member 2;
datatype member 3;
datatype member 4;
-
-
datatype member N;
};
```

Example

```
struct student
{
int rollno;
char sname[20];
Float ~~pag~~ ptage;
};
```

(*) Declaring Structure Variable :

The way we can declare variable and array before using it in program. Similarly we can declare structure variable before using it. Structure variable is the runtime entity of structure. There are two way we can declare structure variable.

① At the time of Structure Definition

Ex.

```
struct student
{
  int rollno;
  char sname[20];
  float ptage;
}s1, s2;
```

Scanned with CamScanner

② After the Structure Definition

Ex.

struct student S1, S2.

(*) Accessing Structure Members

structure members can be accessed and assigned values in a number of ways. structure members have no meaning individually without the structure. In order to assign a value to any structure member, the name must be linked with the structure variable using a dot (.) operator also called period of member access operator.

Ex.-

① Initiallization of structure variable at the time of declaration

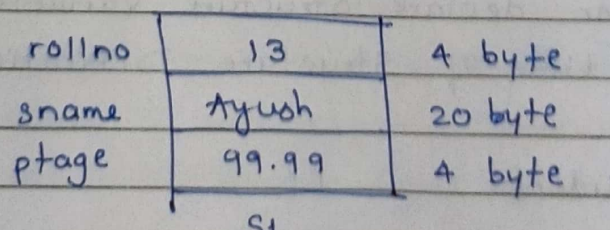
Ex.

```
struct student S1 = {13, "Manoj", 99.99};
```

② Initiallization of structure variable after the declaration.

Ex.-

```
S1.rollno = 13;
strcpy(S1.sname, "Manoj");
S1.m1 = 60;
S1.m2 = 70;
S1.m3 = 80;
```



Scanned with CamScanner

15. WAP to read 2 students details and print it along with total and average.

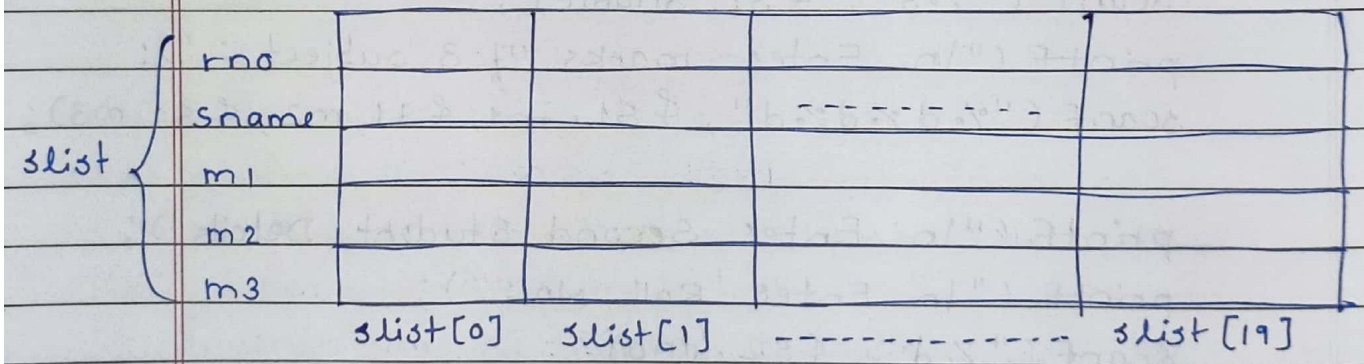
→
```
#include <stdio.h>
struct stud
{
    int rno;
    char sname[20];
    int m1, m2, m3;
};
int main()
{
    struct stud s1, s2;
    int total;
    printf("\n Enter First student details");
    printf("\n Enter Roll No:");
    scanf("%d", &s1.rno);
    printf("\n Enter Name:");
    scanf("%s", &s1.sname);
    printf("\n Enter marks of 3 subject:");
    scanf("%d %d %d", &s1.m1, &s1.m2, &s1.m3);

    printf("\n Enter Second Student Details");
    printf("\n Enter Roll No:");
    scanf("%d", &s2.rno);
    printf("\n Enter Name:");
    scanf("%s", &s1.sname);
    printf("\n Enter marks of 3 subject:");
    scanf("%d %d %d", &s2.m1, &s2.m2, &s2.m3);

    printf("\n First Student Details");
    printf("\n Roll No: %d", s1.rno);
    printf("\n Name: %s", s1.sname);
```

Scanned with CamScanner

```
printf("\n M-I : %d\n M-II : %d\n M-III : %d",
        s1.m1, s1.m2, s1.m3);
total = s1.m1 + s1.m2 + s1.m3;
printf("\n Total : %d", total);
printf("\n Percentage : %.2f", (float) total / 3);
printf("\n **********************");
printf("\n Second Student Details");
printf("\n Roll No : %d", s2.rno);
printf("\n Name : %s", s2.sname);
printf("\n M-I : %d\n M-II : %d\n M-III
: %d", s2.m1, s2.m2, s2.m3);
total = s2.m1 + s2.m2 + s2.m3;
printf("\n Total : %d", total);
printf("\n Percentage : %.2f", (float) total / 3);
}
```

25/01/2022

(*) Array of Structure :



Scanned with CamScanner

152. WAP to read student details & print along with total & average. (using array of structure).

→
```
#include<stdio.h>
struct stud
{
    int rno;
    char sname[20];
    int m1, m2, m3;
};
int main()
{
    struct stud slist[20];
    int i, n, total;
    printf("\n Enter how many details to be read:");
    scanf("%d", &n);
    printf("\n Enter student Details");
    for(i=0; i<n; i++)
    {
        printf("\n Enter Roll No:");
        scanf("%d", &slist[i].rno);
        printf("\n Enter Name:");
        scanf("%s", &slist[i].sname);
        printf("\n Enter marks of 3 subject:");
        scanf("%d%d%d", &slist[i].m1, &slist[i].
        m2, &slist[i].m3);
    }
    printf("\n Roll No  Student Name\t\tM-I
    M-II  M-III  Total  Percentage");
    for(i=0; i<n; i++)
    {
        printf("\n%-8d", slist[i].rno);
```

Scanned with CamScanner

```
        printf("\t%-14s", slist[i].sname);
        printf("\t%-9d%-9d%-9d", slist[i].m1,
        slist[i].m2, slist[i].m3);
        total = slist[i].m1 + slist[i].m2 + slist[i].m3;
        printf("%5d%12.2f", total, (float)total/3);
      }
}.
```

153. WAP with functions read_slist, print_slist to read and print details of students.

→

```
#include <stdio.h>
struct stud
{
    int rno;
    char sname[20];
    int m1, m2, m3;
};
int main()
{
    int n;
    struct stud slist[20];
    printf("\n Enter how many details to be read:");
    scanf("%d", &n);
    void read_slist(struct stud*, int);
    void print_slist(struct stud*, int);
    printf("\n Enter student details :");
    read_slist(slist, n);
    printf("\n********** Student Details *****");
    print_slist(slist, n);
}
void read_slist(struct stud *s, int x)
{
```

Scanned with CamScanner

```
    int i;
    for(i=0; i<x; i++)
    {
    printf("\n Enter Roll No: ");
    scanf("%d", &s[i].rno);
    printf("\n Enter Name : ");
    scanf("%s", &s[i].sname);
    printf("\n Enter marks of 3 subject : ");
    scanf("%d%d%d", &s[i].m1, &s[i].m2, &s[i].m3);
    }
}
void print_slist(struct stud *s, int x)
{
    int i, total;
    printf("\n Roll No      Student Name\t M-I    M-II
     M-III      Total      Percentage");
    for(i=0; i<x; i++)
    {
     printf("\n %-8d", s[i].rno);
     printf("\t %-14s", s[i].sname);
     printf("\t %-9d %-9d %-9d", s[i].m1, s[i].m2,
                                  s[i].m3);
     total = s[i].m1 + s[i].m2 + s[i].m3;
     printf("%5d %12.2f", total, (float) total/3);
    }
}
```

Scanned with CamScanner

"Structure with Pointer" 26/01/2022

- ex.

```
#include <stdio.h>
#include <string.h>
struct stud
{
    int rno;
    char sname[20];
};
int main()
{
    struct stud s1 = {1, "Manoj"};
    void modify data {struct stud};
    printf("\n Roll No: %d", s1.rno};
    printf("\n Name : %s", s1.sname};
    modify data (s1);
    printf("\n Roll No: %d", s1.rno};
    printf("\n Name : %s", s1.sname};
}
void modify data {struct stud *s}
{
    s.rno = 13;
    strcpy(s.name, "Tushar");
}
```

Note ⊛ Changes made on formal parameter will not change on actual parameter 's1'.

154.

→

```
void modifydata (struct stud * s)
{
    /*s->rno=2;
      strcpy(s->sname, "Pramod"); */
    (*s).rno = 12;
    strcpy((*s).sname, "Prakash");
}
```

Scanned with CamScanner

"Union"

Note: (using union the procedure is time consuming)

```
#include<stdio.h>
struct stud
{
        int rno;
        char sname[20];
        float ptage;
};
union ustud
{
        int rno;
        char sname[20];
        float ptage;
};
int main()
{
        struct stud s1;
        union ustud u1;
        printf("\n size = %d", sizeof(s1));
        printf("\n size = %d", sizeof(u1));
}
```

(*) File handling in C:

Sometime we need to store output in a file for permenant storage instead of just displaying it. In C lang. this can be solved by file handling

- Operations on file →

1) creating a new file
2) opening an existing file
3) reading from file
4) writing to a file.

Scanned with CamScanner

5) searching in a file
6) updating a file
7) deleting in a file
8) closing a file

- File handling modes :

1) r → ~~reading~~ open & a existing file for reading purpose.
2) w → create a new file for writing purpose. if file already exist then open it & overwrite it.
3) a → (append mode) → open a text file for writing in append mode if it doesn't exist then a new file is created
4) r+ → file will be open for reading and writing both.
5) w+ → file will be open for reading & writing both it first truncate the file to the zero length.
6) a+ → open a text file for reading & writing both it creates a file if it does not exist. The reading will start from begining but writing will start from append end
7) rb → open a binary file in read mode
8) wb → open a binary file in write mode
9) ab → open a binary file in append mode.

Scanned with CamScanner

classmate
Date
Page

• File handling functions :

1) fopen → this function use to open a file
filepointer = fopen ("fileopen", "mode");
(it supports only txt files)

2) fclose → this function use to close a file.
fclose (filepointer);

3) fputc → writing a character into a file
fputc (char var, filepointer);

4) fgetc → reading single character from a file.
char ch = fgetc (filepointer);

5) fputs → write a string into a file
fputs (char str[], filepointer);
fputs (s, fp);

6) fgets → reading single string from a file.
fgets (string S, int n, fileoperatee)

7) feof () → to locate end of file.
! feof () OR if (fp == NULL)

8) fseek () → to locate the file pointer at a given position
fseek (filepointer, int size, SEEK_SET ← 0 / (OR) / SEEK_CUR ← 1 / (OR) / SEEK_END ← 2)
} constant has values (0, 1, 2)

9) ftell () → to locate the current position.
ftell (filepointer);

Scanned with CamScanner

156. WAP to create a new file & write a data into it.

→

```
#include<stdio.h>
int main ()
{
        char fname [20], ch;
        FILE *fp;
        printf("\n Enter File Name :");
        scanf("%s", &fname);
        fp = fopen (fname, "w");
        if (fp == NULL)
        {
            printf ("\n File Creation/Opening Error");
        }
        else
        {
            printf("\n Enter Data['$']:");
            ch= getchar();
            while (ch != '$')
            {
                fputc(ch, fp);
                ch = getchar();
            }
            printf("\n File Created Successfully");
            fclose (fp);
        }
}
```

157. WAP to open a file & append the data.

```
#include <stdio.h>
int main ()
{
```

Scanned with CamScanner

```
char fname[20], ch;
FILE *fp;
printf("\n Enter File Name : ");
scanf("%s", &fname);
fp = fopen(fname, "a");
if (fp == NULL)
{
    printf("\n File creation/opening Error");
}
else
{
    printf("\n Enter Data ['$']: ");
    ch = getchar();
    while (ch != '$')
    {
        fputc(ch, fp);
        ch = getchar();
    }
    printf("\n File Created Successfully");
    fclose(fp);
}
}
```

158. WAP to create a new file & read a data from it.

→

```
#include<stdio.h>
int main()
{
    char fname[20], ch;
    FILE *fp;
    printf("\n Enter File Name : ");
    scanf("%s", &fname);
```

Scanned with CamScanner

```
fp = fopen (fname, "r");
if (fp == NULL)
{
    printf ("\n File Creation/Opening Error");
}
else
{
    printf ("\n Data From File: \n");
    while (!feof (fp))
    {
        ch = fgetc (fp);
        putchar (ch);
    }
    fclose (fp);
}
}
```

159. WAP to count words & line from a file.

→

```
#include <stdio.h>
int main ()
{
    char fname [20], ch;
    FILE *fp;
    int wcnt = 0, lcnt = 0;
    printf ("\n Enter File Name :");
    scanf ("%s", &fname);
    fp = fopen (fname, "r");
    if (fp == NULL)
    {
        printf ("\n File Opening Error");
    }
    else
    {
```

Scanned with CamScanner

```
do
{
    ch = fgetc (fp);
    if (ch==' ' || ch=='\n')
    {
        wcnt ++ ;
    }
    if (ch=='\n')
        lcnt++;
} while (ch != EOF);
printf("\n word count : %d", wcnt + 1);
printf ("\n Line count : %d", lcnt + 1);
fclose (fp);
}
}
```

160. WAP to calculate the size of given file.

161. WAP to print the content of file in reverse order.

160. →

```
#include<stdio.h>
int main ()
{
    char fname [20], ch;
    FILE *fp;
    int l;
    printf ("\n Enter File Name : ");
    scanf ("%s", &fname);
    fp = fopen (fname, "r");
    if (fp == NULL)
    {
        printf("\n File Opening Error");
    }
```

Scanned with CamScanner

```
    else
    {
        fseek (fp, 0, SEEK_END);
        l = ftell (fp);
        printf("\n File size : %d byte", l);
        fclose (fp);
    }
}
```

161. →

```
#include<stdio.h>
int main()
{
    char fname[20], ch;
    FILE *fp;
    int l;
    printf("\n Enter File Name :");
    scanf("%s", &fname);
    fp = fopen(fname, "r");
    if (fp==NULL)
    {
        printf("\n File Opening Error");
    }
    else
    {
        fseek (fp, 0, SEEK_END);
        l = ftell (fp);
        while (i<l)
        {
            i++
            fseek (fp, -i, SEEK_END);
            printf("%c", fgetc (fp)); OR
            putchar (fgetc (fp));
```

Scanned with CamScanner

continue

```
    }
    fclose(fp);
}
```

⊙ file handling functions : (continued here)

10) fprintf() → write data into a file.

fprintf(filepointer, "control string", argument list);

ex. → int x = 5;

fprintf(filepointer, "%d", x)

11) fscanf() → read data into a file.

Fscanf(filepointer, "control string", argument list);

162. WAP to ~~print~~ write integer value into a file

→

```
#include<stdio.h>
int main()
{
    char fname[20], ch;
    FILE *fp;
    int x, n;
    printf("\n Enter File Name :");
    scanf("%s", &fname);
    fp = fopen(fname, "w");
    if(fp == NULL)
    {
        printf("\n File Creation/opening Error");
    }
    else
    {
        printf("\n Enter how many no. to be read :");
        scanf("%d", &n);
        while(n != 0)
        {
```

Scanned with CamScanner

```
            printf("\n Enter Number : ");
            scanf("%d", &x);
            fprintf(fp, "%d", x);
            n--;
        }
        fclose (fp);
    }
}
```

163. WAP to read integer value into a file & count even, odd from it

→

```
#include <stdio.h>
int main ()
{
    char fname [20], ch;
    FILE *fp;
    int n, ecnt = 0, ocnt = 0;
    printf("\n Enter a file name : ");
    scanf ("%s", & fname);
    fp = fopen (fname, "r");
    if ( fp == NULL)
    {
        printf ("\n File Opening Error");
    }
    else
    {
        while ( ! feof (fp))
        {
            fscanf (fp, "%d", &n);
            printf ("%4d", n);
            if (n%2 == 0)
                ecnt ++ ;
            else
                ocnt ++;
```

Scanned with CamScanner

```
    }
    printf ("\n Even Count = %d", ecnt);
    printf("\n Odd Count = %d", ocnt);
  }
  fclose (fp);
  }
}
```

12) rewind() → it moves file pointer at begining of file.

```
rewind (file pointer);
```

164. WAP to ~~create~~ open existing file and add new data from the begining of file.

→

```
#include<stdio.h>
int main ()
{
    char fname [20], ch;
    FILE *fp;
    printf ("\n Enter File Name :");
    scanf ("%s", &fname);
    fp = fopen (fname, "r+");
    if (fp == NULL)
    {
        printf ("\n File Creation/Opening Error");
    }
    else
    {
        rewind (fp);
        printf ("\n Enter New Data:");
        ch = getchar ()
        while (ch != '$')
        {
```

Scanned with CamScanner

29/01/2022

```
        fputc (ch, fp);
        ch = getchar ();
    }
    fclose (fp);
  }
}
    while (!feof (fp))
    {
     putchar (fgetc(fp));
    }
    rewind (fp);
    printf ("\n After Rewind :\n");
    while (!feof (fp)) }
    {
     putchar (fgetc(fp));
    }
{  }}
   fclose(fp);
}
```

13)

fwrite() → this funct^n use to write any type of value into file.

- syntax fwrite → fwrite (object, sizeof(object), count of element, file pointer);

165. WAP to print array using function fwrite.

→

```
#include<stdio.h>
int main ()
{
    char fname[20];
    FILE *fp ;
```

Scanned with CamScanner

```
printf("\n Enter File Name:");
scanf("%s", &fname);
fp = fopen(fname, "w");
if(fp == NULL)
{
    printf("\n File Creation Error");
}
else
{
    int arr[5] = {10, 20, 30, 40, 50};
    fwrite(arr, sizeof(int), 5, fp);
    if(fwrite != 0)
    {
        printf("\n Content write successfully");
    }
    else
    {
        printf("\n Content not write");
    }
}
fclose(fp);
}
```

14) fread() → this funct^n use to read any type of value ~~into~~ from file.

- syntax → fread(object, sizeof(object), count of element, fp);

15) remove() → this funct^n use to remove the file.

- syntax → remove(fp);

Scanned with CamScanner

166. WAP to read array using funct^n fread from a file.

→
```
#include < stdio.h>
int main ()
{
    int arr[5] , i ;
    char fname [20];
    FILE *fp ;
    printf("\n Enter File Name : ");
    scanf ("%s", & fname) ;
    fp = fopen ( fname , "r") ;
    if (fp = NULL)
    {
        printf("\n File Creation Error ");
    }
    else
    {
        fread (arr , sizeof (int) , 5 , fp) ;
        for (i=0 ; i<5 ; i++)
        {
            printf("%4d", arr [i]) ;
        }
    }
    fclose (fp) ;
}
```

167. WAP to remove file.

→
```
#include < stdio.h>
int main ()
{
    int arr[5], i;
    char fname [20];
```

Scanned with CamScanner

classmate
Date
Page

```
FILE *fp;
printf("\n Enter File Name:");
scanf("%s",&fname);
if(!remove(fname))
    printf("\n File Deleted sucessfully");
else
    printf("\n File Deletion Error");
}
```

Scanned with CamScanner