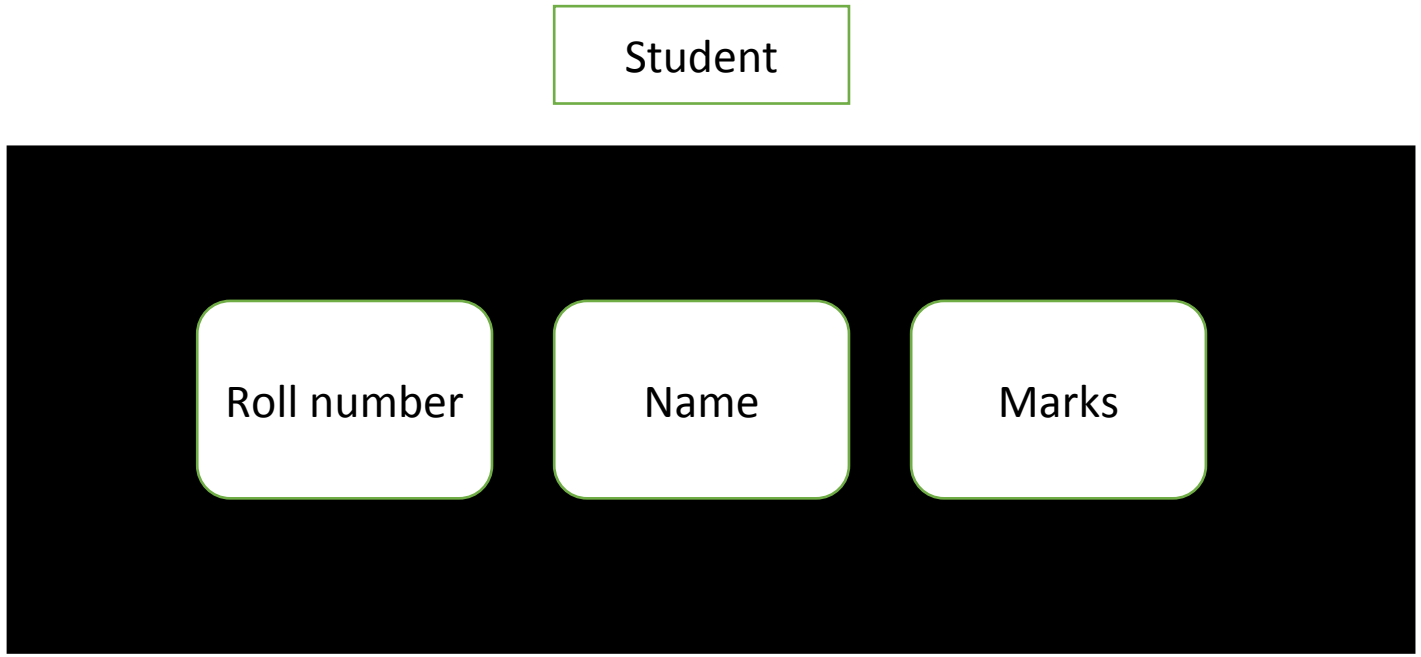
Structures



A structure by name **Student**

User defined data type

Some call a structure as ‘derived data type’

Structures

```
struct student
{
    int roll;
    char *name;
    float marks;
};
```

A structure by name **Student**

struct keyword

Semicolon at the end

Structure members

Structures

```c
#include<stdio.h>
// Declaring a Structure
struct student
{
    int roll;
    char *name;
    float marks;
};
int main()
{
    struct student s1;
}
```

Structure declared outside main function

Structure variable s1 declared

Study the syntax of declaring a structure variable

Structure Variables and DOT operator

```c
#include<stdio.h>
// Declaring a Structure
struct student
{
    int roll;
    char *name;
    float marks;
};
int main()
{
    struct student s1;
    //Use the . operator
    s1.roll = 10;
    printf("%d",s1.roll);
    return 0;
}
```

DOT operator (.)

Accessing members of a structure using the DOT operator and Structure variable

Homework 1 : Initialize the structure member 'marks'

Multiple structure variables

```c
#include<stdio.h>
// Declaring a Structure
struct student
{
    int roll;
    char *name;
    float marks;
};
int main()
{
    struct student s1,s2;
    //Use the . operator
    s1.roll = 10;
    s1.name = "Nivedita";
    s1.marks = 49.5;

    s2.roll = 20;
    s2.name = "Dravid";
    s2.marks = 45;

    printf("%s \t %s",s1.name,s2.name);
    return 0;
}
```

Multiple structure variables

Notice – How a pointer to character variables is also accessed

Homework 2 : Type this program, but give your own data for structure members

Multiple structures

```c
#include<stdio.h>
struct student
{
    int roll;
    char *name;
    float marks;
};
struct subject
{
    char *name;
    int numerical_code;
};
int main()
{
    struct student s1,s2;
    s1.numerical_code = 10;
    printf("%d",s1.numerical_code);
    return 0;
}
```

Multiple structures

Type the code given and run it

Homework 3: Study the output and write down 'why' of the output

Multiple structures

```c
#include<stdio.h>
struct student
{
    int roll;
    char *name;
    float marks;
};
struct subject
{
    char *name;
    int numerical_code;
};
int main()
{
    struct student s1,s2;
    s1.numerical_code = 10;
    printf("%d",s1.numerical_code);
    return 0;
}
```

Multiple structures

Type the code given and run it

Homework 4: Modify the code to access the 'numerical_code' member

Memory occupied by a structure

```c
#include<stdio.h>
struct student
{
    int roll;
    char *name;
    float marks;
};
struct subject
{
    int numerical_code;
    float marks;
};
int main()
{
    printf("%d\n",sizeof(struct student));
    printf("%d\n",sizeof(struct subject));
    return 0;
}
```

How much memory does a structure occupy ?

Type the code given and run it

Homework 5: The output can be surprising

Will a few structure variables be enough ?

```c
#include<stdio.h>
// Declaring a Structure
struct student
{
    int roll;
    char *name;
    float marks;
};
int main()
{
    struct student s1,s2;
    //Use the . operator
    s1.roll = 10;
    s1.name = "Nivedita";
    s1.marks = 49.5;

    s2.roll = 20;
    s2.name = "Dravid";
    s2.marks = 45;

    printf("%s \t %s",s1.name,s2.name);
    return 0;
}
```

A rewind

How will you enter data for 50 students?

Create 50 structure variables?

Array of structure variables

```c
#include<stdio.h>
// Declaring a Structure
struct student
{
    int roll;
    char *name;
    float marks;
};
int main()
{
    struct student s1,s2;
    //Use the . operator
    s1.roll = 10;
    s1.name = "Nivedita";
    s1.marks = 49.5;

    s2.roll = 20;
    s2.name = "Dravid";
    s2.marks = 45;

    printf("%s \t %s",s1.name,s2.name);
    return 0;
}
```

A rewind

How will you enter data for 50 students?

An array of structure variables

Array of structure variables

```c
#include<stdio.h>
struct student
{   int roll;
    char name[50];
    float marks;
};
int main()
{
    int i = 0;
    struct student s[3];
    for(i=0;i<3;i++)
    {
        printf("Enter roll no: ");
        [redacted]
        printf("Enter first name: ");
        scanf("%s",s[i].name);
        printf("Enter marks: ");
        [redacted]
    }
    for(i=0;i<3;i++)
    {
        printf("Roll no is:%d \t",[redacted]);
        printf("First name is: %s\t",s[i].name);
        printf("Mark is: %f \n",[redacted]);
    }
    return 0;
}
```

Read user input

Array of Structures

Homework 6: Complete the code

Find the output

```c
#include<stdio.h>
struct student
{
    int roll=20;
    char name[50];
    float marks;
};
int main()
{
    struct student s1;
    printf("%d",s1.roll);
    return 0;
}
```

Find the output

'roll' is a structure member which has been initialized with the value 20

Homework 7: Run the code and reflect on the result

Pointer to a structure

```c
#include<stdio.h>
struct student
{
    int roll;
};
int main()
{
    struct student s1;
    //Declaring a pointer to a structure
    struct student *ptr;
    //ptr is a pointer that points to 'student'
    ptr = &s1;
    ptr->roll = 10;
    printf("%d",ptr->roll);
    return 0;
}
```

-> operator is used to access members using a pointer

ptr is a pointer that points to a structure variable

Homework 8: Run the code

Pointer to a structure

```
1  #include<stdio.h>
2  struct student
3  {
4      int roll;
5  };
6  int main()
7  {
8      struct student s1={20};
9      //Declaring a pointer to a structure
10     struct student *ptr;
11     //ptr is a pointer that points to 'student'
12     ptr = &s1;
13     printf("%d",ptr->roll);
14     return 0;
15 }
```

WHY do we need a pointer to a structure ?

Another method to initialize structure members

Line number 8 is another way to initialize structure members

Homework 8: Run the code

We will come back to this later

How do you know the amount of memory required by a user ? - Necessity of DMA

```c
#include<stdio.h>
int main()
{
    char a[10];
    char b[10];
    //Reading a string via scanf
    scanf("%s",a);
    getchar();
    printf("%s\n",a);
    //Reading a string via fgets
    fgets(b,10,stdin);
    puts(b);
    return 0;
}
```

Static memory allocation

Memory is allocated and freed implicitly

Homework 9: Run the code (Give the input such that the string length >> 10)

Dymanic Memory Allocation

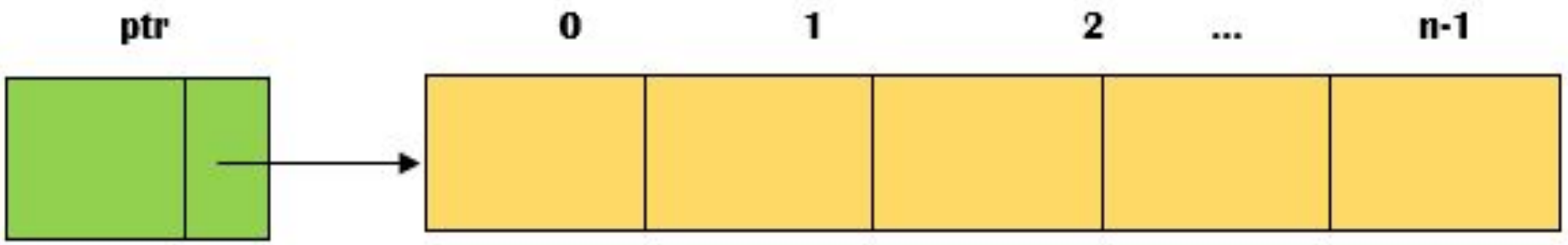
- malloc - (cast-type*)malloc(n* sizeof(type))
- calloc - (cast-type*)calloc(n,sizeof(type))
- realloc- (cast-type*)realloc(ptr, n*sizeof(type))
- free

Functions for DMA

Memory has to be allocated and freed explicitly

stdlib.h

Dymanic Memory Allocation - malloc




```
printf("Enter max: ");
scanf("%d",&n);
int *ptr;
ptr = (int*)malloc(n*sizeof(int));
```

If n = 5, 5 blocks of 4 bytes each = 20 bytes of memory is dynamically allocated

Consecutive memory locations (a memory block)

Dymanic Memory Allocation - malloc

```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i = 0, n=0;
    printf("Enter max: ");
    scanf("%d",&n);
    int *ptr;
    ptr = (int*)malloc(n*sizeof(int));
    for (i = 0; i < n; ++i)
    {
        ptr[i] =  i;
    }

    printf("The elements of the array are: ");
    for (i = 0; i < n; ++i)
    {
        printf("%d, ", ptr[i]);
    }
    return 0;
}
```

malloc in code

Study line number 9 in detail (explanation provided in notes)

Homework 10

Is this correct ?

```c
#include<stdio.h>
int main()
{
    int i = 0, n=0;
    printf("Enter number: ");
    scanf("%d",&n);
    int a[n];
    for(i=0;i<n;i++)
    {
        a[i]=i;
    }
    for(i=0;i<n;i++)
    {
        printf("%d,",a[i]);
    }
return 0;
}
```

Is this correct ?

This code runs perfectly ?

Homework 11

Linked list

Assignment - 1

LL.c is the program name (uploaded in teams)

Comment each line of code

Draw and explain the list formation

Assignment

- Create a C program that would read the academic record of 5 students. After the record of all the students have been entered, the user should be provided with a menu which has the following:
  - Enter ‘1’ to display the names of all the students
  - Enter ‘2’ to find the average mark of all the students in Maths
  - Enter ‘3’ to search for a particular student based on the roll number

  Each record should contain the following:
  - Name of the student
  - Roll no
  - Mark in Maths
  - Mark in Sanskrit
  - Mark in Programming

  The code to display the name of all the students, finding the average and to search should be within three separate functions (one function to display the name of all students, one function for finding the average, likewise one function to search for a student based on a roll number)