C++ Notes Day-3 Date: 07 June 2025

Lets revise

- Pointers
- Constant qualifier
- Constant and Pointer Combination
    - int *ptrNum
    - Here, ptrNum is non-constant int pointer variable which is ready to store the address non-constant int variable.

```
int main()
{
    int *ptrNum;
    int Num1=100;
    ptrNum=&Num1;   //Assiging address of non-constant int variable to non-contant pointer of type int

    printf("\n The Value of Num1 through ptr:%d",*ptrNum);  //Dereferncing
    printf("\n The Address of Num1 through ptr:%p", ptrNum);

}
```

    - const int *ptrNum / int const *ptrNum / const int const *ptrNum
    - Rule: if const keyword is present before the start (*) it means ptr is looking for the constant variable.

```
#include <iostream>
using namespace std;
int main()
{
    const int *ptrNum;  //Here, ptrNum is non-constant integer pointer variable which is ready to store the address of constant integer variable

    const int Num=100;  //

    ptrNum=&Num;

    printf("\n The Value of Num through ptr:%d", *ptrNum);
    printf("\n The Address of Num through ptr:%p", ptrNum);

    //*ptrNum=200;  //NOT OK, Num is constant, error: assignment of read-only location '* ptrNum'

    const int Num1=200;
    ptrNum=&Num1;
    printf("\n The Value of Num through ptr:%d", *ptrNum);
    printf("\n The Address of Num through ptr:%p", ptrNum);
```

```
    return 0;
}
```

- int *const ptrNum
- Here, ptrNum is constant pointer of type integer and ready to store the address of any non-constant int variable.

```
int main()
{
    int Num=100;
    int *const ptrNum=&Num;   //ptrNum is constant pointer int variable
which storing the address of non-constant int variable Num.

    *ptrNum=300;    //
    ptrNum=&Num2;   //
}
```

- const int *const ptrNum: copy the example and code from program
- int *ptrNum const: Not Valid
- Type Casting in pointers

```
int main()
{

    const int Num1=100;
    const int *ptrNum1=&Num1;

    //*ptrNum1=300; //NOT OK

    printf("Value of Num1:%d\n",Num1);      //100
    printf("Value of Num1 using Dref:%d\n",*ptrNum1);   //100

    int *ptrNum2=(int*)&Num1;
    *ptrNum2=400;   //OK
    printf("Value of Num1:%d\n",Num1);      //100
    printf("Value of Num1 using Dref:%d\n",*ptrNum2);   //400   //Un-
Expected Behavior

    return 0;
}
```

Structure in C

- If I need single data value inside a program I can use basic data type.
- If I need multiple data values of similer type inside the program i can use dervied data type i.e. Arrays.
- If I need multiple data valaues of different data types under single name or unit then I can use struct or union.

- AccuontNo-int, CustomerName-char, Age-int, Balance-float:- BankAccount
- EmpId-int, EmpName-char, EmpSalary-float, EmpDesignation-char: Employee
- Strcuture is dervied data type in C/C++. It is known as user defined data type.
- We can declara structure indise any method it is known as local structure.
- Structure is set of different variables.
- struct is keyword in C/C++ to declare the structure.
- In case of local structure we can't define its object globally.
- Synatx to declare Strructure

```
struct Employee
{

};
```

 Image

- Data members of the structure gets space inside the structure object.
- To access data members of the structure with object we use Member Selection operator (.).
- We can also declare pointer variable of the structure. To access structure data members with pointer variable we must use (->) arrow.
- passing structure object as Address

```
#include <iostream>
using namespace std;

//void ScanData(Employee);   //NOT OK

struct Employee         //Declaration of the structure Employee
{
    int EmpId;          //Data Member of the structure Employee
    char Name[50];      //Data Member of the structure Employee
    float Salary;       //Data Member of the structure Employee
    int Age;            //Data Member of the structure Employee
};

void ScanData(Employee *);      //Declaration of Global Method
void PrintData(Employee *);     //Declaration of Global Method


int main()
{
    struct Employee emp1;   //emp1 is a variable of type Employee

    ScanData(&emp1);        //Call By Address
    PrintData(&emp1);

    return 0;
}
```

```
void ScanData(Employee *emp1)           //Global Method
{
        printf("Enter Name: ");
        fflush(stdout);
        scanf("%s", emp1->Name);
        printf("\nEnter EmpId: ");
        fflush(stdout);
        scanf("%d", &emp1->EmpId);
        printf("\nEnter Age: ");
        fflush(stdout);
        scanf("%d", &emp1->Age);
        printf("\nEnter Salary: ");
        fflush(stdout);
        scanf("%f", &emp1->Salary);
}

void PrintData(Employee *emp1)      //Global Method
{
    printf("\n The Values of Employee:");
    printf("\n Name: %s", emp1->Name);
    printf("\n Age: %d", emp1->Age);
    printf("\n Salary: %f", emp1->Salary);
    printf("\n EmpId: %d", emp1->EmpId);
}
```

- Limitation of C
  - In C data manupulation is done through global methods. It means any global method can access any global data. So data security is difficult to achieve.
  - There is no string data type in C.
  - Due to presenace of large no. of global methods code clarity is difficult.

Introduction to C++

- History
  - Inventor: Bjarne Stroustup
  - Year: 1979
  - Where: At L&AT Bell Lab
  - Initial Name: C with Classes
  - Renamed in 1983 as C++
  - C++ is being staderedized by ISO Working Group
  - Standereds of C++
    - C++ 98
    - C++ 03
    - C++ 11
    - C++ 14
    - C++ 17
    - C++ 20
    - C++ 23
    - C++ 26

- C++ is high level object oriented programming language.
- Its follow botton up approach.
- C++ having all the features of C, hence it is also known as Hybrid Programming Language
- Like C, C++ is also statically type check as well as strongly type check programming language
- Datatypes in C++
    - Basic Datatypes
        - int
        - char
        - float
        - double
        - bool
        - void
        - wchar_t (typedef unsigned short wchar_t)
    - Derived Data Types
        - Arrays
        - Pointers
        - Functions
        - Referennces
    - User Defined Data Types
        - Strcuture
        - Union
        - Classes
    - Type Modifiers
        - short
        - long
        - signed
        - unsigned
    - Type Qualifiers
        - const
        - volatile
    - Execution flow C++
        - Bjarne Stroustup invented a system software named as 'cfront'. It is a interpreter which used to translate the C++ code into C.
    - Access Modifiers
        - To control the visibility or access of the data in C++ the following Access Modifiers are used
            - public
            - private
            - protected
- Structure in C++
    - We can defined member functions inside the C++ Structure
    - To create object of the struct data type struct keyword is optional.
    - By default structure data memebrs are public.
    - Data Member
        - Variables decalred inside the structire are known as Data Members.
        - These are also known as Fields, Property, attributes

- Member Function
  - A Function defined inside the structure body is known as Member Function of the structure.
  - Member Function is also known as Method/Procedure/Behaviour/Message.
  - A member function with bdoy is known as Concrete function.
  - A member function without bosy is known as Abstract Function.
- Example:

```
struct Employee
{
    int EmpId;          //Data Member or Field or Property or Attribute
    char Name[50];

    void SetData()      //Memeber Function or Method or Procedure
    {

    }
}emp;                   // emp is object or instance or variable of
type Employee
```

- Example:

```
#include <iostream>
using namespace std;
struct Employee         //Declaration of the structure Employee
{
    int EmpId;          //Data Member of the structure Employee
    char Name[50];      //Data Member of the structure Employee
    float Salary;       //Data Member of the structure Employee
    int Age;            //Data Member of the structure Employee

    void ScanData()
    {
            printf("Enter Name: ");
            fflush(stdout);
            scanf("%s", Name);
            printf("\nEnter EmpId: ");
            fflush(stdout);
            scanf("%d", &EmpId);
            printf("\nEnter Age: ");
            fflush(stdout);
            scanf("%d", &Age);
            printf("\nEnter Salary: ");
            fflush(stdout);
            scanf("%f", &Salary);
    }

    void PrintData()        //Member Function of struct Employee
    {
```

```
        printf("\n The Values of Employee:");
        printf("\n Name: %s",Name);
        printf("\n Age: %d", Age);
        printf("\n Salary: %f", Salary);
        printf("\n EmpId: %d",EmpId);
    }
};

int main()
{
    struct Employee emp1;    //emp1 is a variable of type Employee

    emp1.ScanData();         //Calling structure method with the help of
structure object using member selection operator (.)

    emp1.PrintData();        //Message Passing

    return 0;
}
```

- use of typedef in structure
  - To give alias name to the structure defined.

```
#include <iostream>
using namespace std;
typedef struct Employee          //Declaration of the structure
Employee
{
    int EmpId;           //Data Member of the structure Employee
    char Name[50];       //Data Member of the structure Employee
    float Salary;        //Data Member of the structure Employee
    int Age;             //Data Member of the structure Employee

    void ScanData()
    {
            printf("Enter Name: ");
            fflush(stdout);
            scanf("%s", Name);
            printf("\nEnter EmpId: ");
            fflush(stdout);
            scanf("%d", &EmpId);
            printf("\nEnter Age: ");
            fflush(stdout);
            scanf("%d", &Age);
            printf("\nEnter Salary: ");
            fflush(stdout);
            scanf("%f", &Salary);
    }

    void PrintData()        //Member Function of struct Employee
    {
```

```
        printf("\n The Values of Employee:");
        printf("\n Name: %s",Name);
        printf("\n Age: %d", Age);
        printf("\n Salary: %f", Salary);
        printf("\n EmpId: %d",EmpId);
    }
}Malkeet;


int main()
{
    Malkeet m1;
    m1.ScanData();
    m1.PrintData();

    return 0;
}
```

- Array of objects of a structures

```
#include <iostream>
#include <vector>
using namespace std;
typedef struct Employee         //Declaration of the structure
Employee
{
    int EmpId;          //Data Member of the structure Employee
    char Name[50];      //Data Member of the structure Employee
    float Salary;       //Data Member of the structure Employee
    int Age;            //Data Member of the structure Employee

    void ScanData()
    {
            printf("Enter Name: ");
            fflush(stdout);
            scanf("%s", Name);
            printf("\nEnter EmpId: ");
            fflush(stdout);
            scanf("%d", &EmpId);
            printf("\nEnter Age: ");
            fflush(stdout);
            scanf("%d", &Age);
            printf("\nEnter Salary: ");
            fflush(stdout);
            scanf("%f", &Salary);
    }

    void PrintData()        //Member Function of struct Employee
    {
        printf("\n The Values of Employee:");
        printf("\n Name: %s",Name);
        printf("\n Age: %d", Age);
```

```
        printf("\n Salary: %f", Salary);
        printf("\n EmpId: %d",EmpId);
    }
}Malkeet;


int main()
{
    Employee arr[10];                   //Array of objects

    for(Employee e: arr)                //Initializing the records
    {
        e.ScanData();
    }


    for(Employee e: arr)                //Printing the records
    {
        e.PrintData();
    }
    return 0;
}
```

- Dynamically created object in structure

```
#include <iostream>
using namespace std;
typedef struct Employee         //Declaration of the structure
Employee
{
    int EmpId;          //Data Member of the structure Employee
    char Name[50];      //Data Member of the structure Employee
    float Salary;       //Data Member of the structure Employee
    int Age;            //Data Member of the structure Employee

    void ScanData()
    {
            printf("Enter Name: ");
            fflush(stdout);
            scanf("%s", Name);
            printf("\nEnter EmpId: ");
            fflush(stdout);
            scanf("%d", &EmpId);
            printf("\nEnter Age: ");
            fflush(stdout);
            scanf("%d", &Age);
            printf("\nEnter Salary: ");
            fflush(stdout);
            scanf("%f", &Salary);
    }

    void PrintData()        //Member Function of struct Employee
```

```
    {
        printf("\n The Values of Employee:");
        printf("\n Name: %s",Name);
        printf("\n Age: %d", Age);
        printf("\n Salary: %f", Salary);
        printf("\n EmpId: %d",EmpId);
    }
}Malkeet;


int main()
{
    Malkeet m1;     //m1 is local variable of type Employee and
gets memory inside function's stack frame also know as statically
defined object
    m1.ScanData();
    m1.PrintData();

    Employee *ptr=(Employee*)malloc(sizeof(Employee));      //NOT
OK, error: invalid conversion from 'void*' to 'Employee*'

    ptr->ScanData();
    ptr->PrintData();

    free(ptr);          //Here memory assigned to ptr is being
deleted

    ptr=nullptr;

    return 0;
}
```

To be disscussed tomorrow (08-06-2025)

Class, object and related c concepts

- Multiple demo
- Header Guard
- #include<abc.h> versus #include"abc.h"
- Namespace

Stream concept

- Standard stream objects associated with console.
- std namespace
- cin, cout, cerr and clog objects

Object Oriented Concepts

- class and object concepts
- characteristics of object.