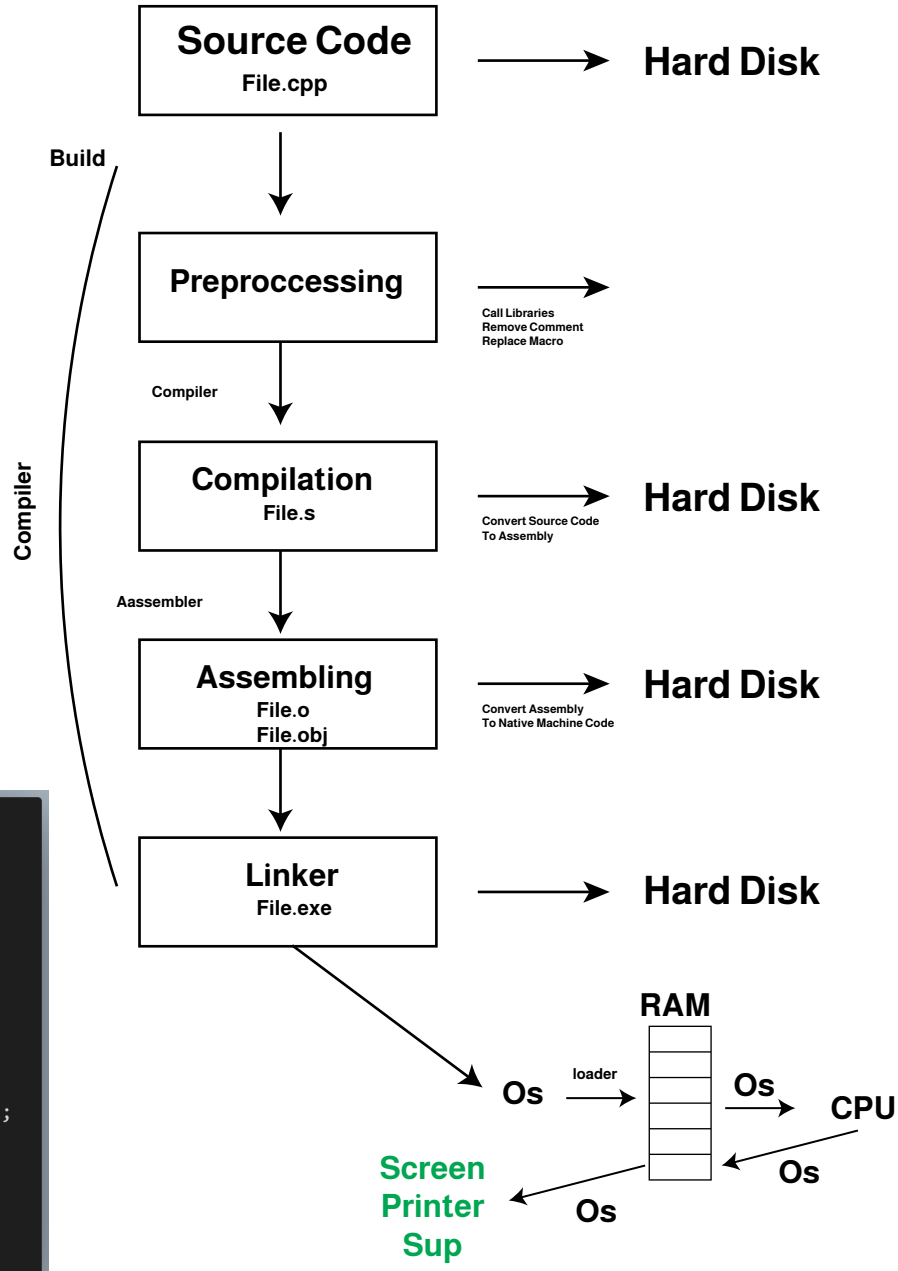
Day1_OOP_Using CPP

Day1
All Concept That We Study Using C --> CPP
Function Overload
Default Parameter
Print,Scan
Function Signture
Class

Encapsulation
Polymorphism
Inheritance
Abstraction




```cpp
//Preproccessing
#include <iostream>
#define PI 3.14
//Global Scope

int main(){//Entry Point


//Local Scope
return 0;//End Of Program
}
```

```cpp
int age;
string Name;
char Gender;

cout<<"Please Enter Your Name"<<endl;
cin>>Name;
_flushall();
cout<<"Please Enter Your Age"<<endl;
cin>>age;
cout<<"Please Enter Your Gender"<<endl;
cin>>Gender;

cout<<"Your Name Is "<<Name<<endl;
cout<<"Your Age Is "<<age<<endl;
cout<<"Your Gender Is "<<Gender<<endl;
```

Data Types

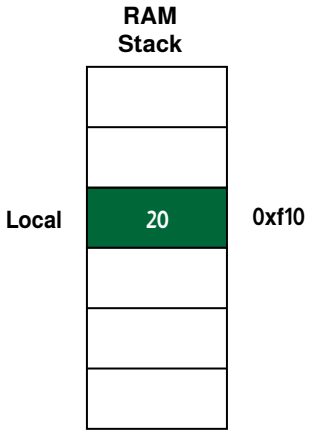
int --> 4
double --> 8
float --> 4
char --> 1
string --> ?
bool --> 1Byte
Array

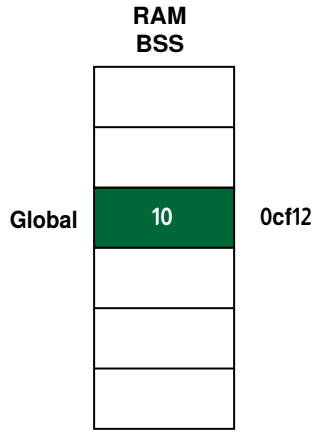
Enum
Struct
Class

Variable

Data Type --> Size
Name
Value
Address
Scope
Life Time

```
#include <iostream>

using namespace std;
        int Global = 10;

int main()
{
            int Local = 20;

  return 0;
}
```





Control Flow

Conditaion Statment

- if
- else
- switch (int , char , enum, bool)

Iteration

- Open Loop
  - While
  - Do While
- Close Loop
  - For

```cpp
#include <iostream>

using namespace std;

int main()
{
    char Gender;
    cout<<"Please Enter Your Gender"<<endl;
    cin>>Gender;
    if(Gender == 'm'){
        cout<<"Gender Is Male"<<endl;
    }
    if(Gender =='f'){
        cout<<"Gender Is Female"<<endl;
    }
    if(Gender == 'm'){
        cout<<"Gender Is Male"<<endl;
    }else{
        cout<<"Gender Is Female"<<endl;
    }

    if(Gender == 'm')
        cout<<"Gender Is Male"<<endl;
    else
        cout<<"Gender Is Female"<<endl;

    int Day = 6;
    if(Day ==1){
        cout<<""<<endl;
    }else if(Day == 2){
     cout<<""<<endl;
    }else{
     cout<<""<<endl;
    }

    return 0;
}
```

Out

```cpp
bool Status = true;
switch(Status){
case true:
    //logic
    break;
case false:
    //logic
    break;
default:
    //logic
    break;
}
```

Out

```cpp
char Color ;
cout<<"Please Enter Color"<<endl;
cin>Color;
switch(Color){
case 'R':
case 'r':
case '99':
    //logic
    break;;
case 'g':
    //logic
case 'y':
    //logic
    break;
default:
cout<<"Error Data Entered"<<endl;
    break;
}
```

```cpp
int Age;
cout<<"Please Enter Your Age"<<endl;
cin>>Age;
while(Age<18 || Age>60){
cout<<"Please Enter Your Age"<<endl;
cin>>Age;
}
cout<<"Your Age Is "<<Age<<endl;

do{
    cout<<"Please Enter Your Age"<<endl;
    cin>>Age;
}while(Age<18 || Age>60);
```

“\n” vs endl

String Is A Immutable Data Type

```cpp
cout<<"Hello My Name Is Mustafa\n";
cout<<"Hello My Age Is 28\n";
cout<<"Hello My Gender Is Male\n";
cout<<"Hello My City Is Arish\n";
cout<<"Hello My Salary Is 1200\n";
```

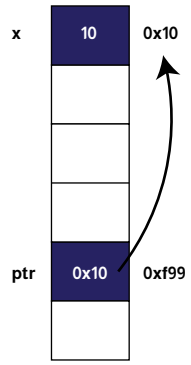

```cpp
cout<<"Hello My Name Is Mustafa"<<endl;
cout<<"Hello My Age Is 28"<<endl;
```

endl هى افضل فى الاستخدام

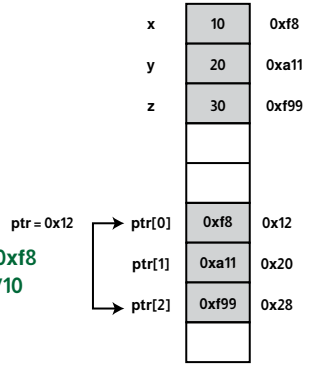
Pointer --> Data Type Can Hold Address

8Byte

```cpp
#include <iostream>

using namespace std;

int main()
{
  //Pointer
  int *ptr;
  int x = 10;
  ptr = &x;
  cout<<ptr<<"\n";//0x10
  cout<<*ptr<<endl;//10
  cout<<*(0x10)<<endl;//10
  cout<<&ptr<<endl;//0xf99
  return 0;
}
```




```cpp
#include <iostream>

using namespace std;

int main()
{
        int *ptr[3];
        int x =10;
        int y = 20;
        int z =30;
        ptr[0] = &x;
        ptr[1] = &y;
        ptr[2] = &z;
        cout<<ptr[0]<<endl;//0xf8
        cout<<*ptr[0]<<endl;//10

  return 0;
}
```

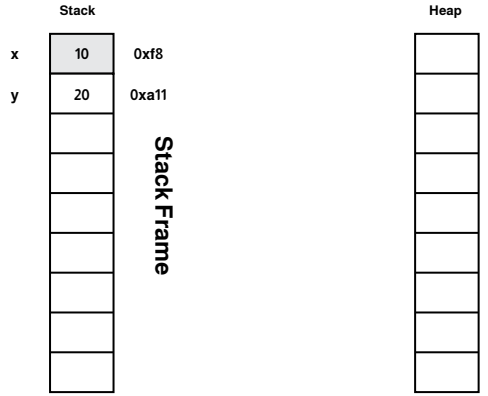


Dynamic Allocation

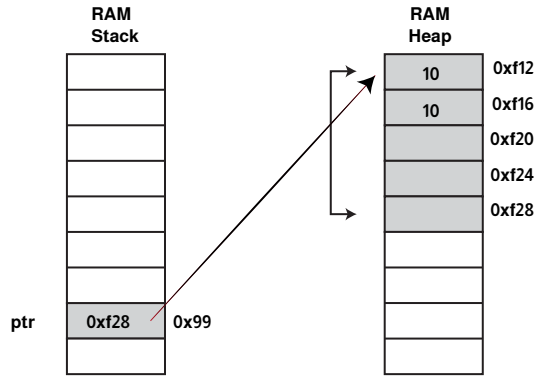
```cpp
#include <iostream>

using namespace std;

int main()
{
        int x =10;
        int y=20;

  return 0;
}
```

OS هو المسؤول عن كل العمليات هنا

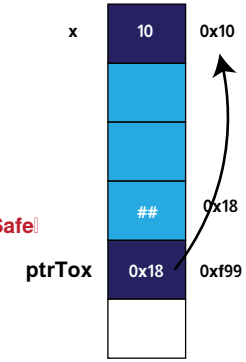
انت المتحكم فى عملية حجز هذه المساحة
انت كمان المسؤول عن عملية تدير او Free للمساحة ده



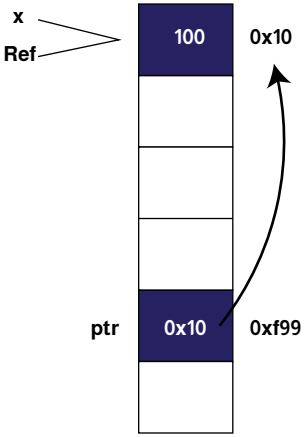
```
#include <iostream>

using namespace std;

int main()
{
        int *ptr;
        ptr = new int[5];
        //ptr+i

  return 0;
}
```




```
int x =10;
int *ptrTox=&x;
ptrTox++;
ptrTox++;
ptrTox++;
*ptrTox = 123;
//Pointer Is UnSafe
```

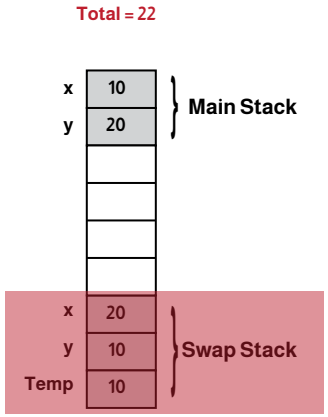



```
#include <iostream>

using namespace std;

int main()
{
  //Alies
  //NeckName
  //Ref
  int x = 10;//Byte
  int *ptr = &x;//8Byte
  int &Ref = x;
  Ref =100;
  return 0;
}
```

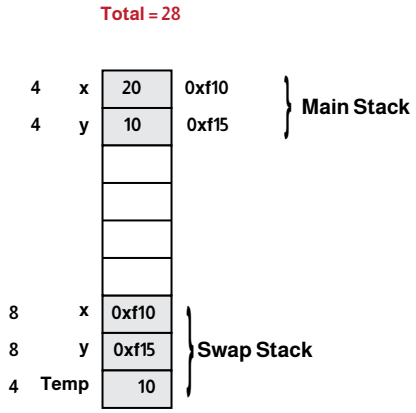


Ref هى تعتبر اسماء مستعارة Alias لمتغيرات متواجدة بالفعل، ولا تقوم بحجز اى مساحة إضافية فى الذاكرة

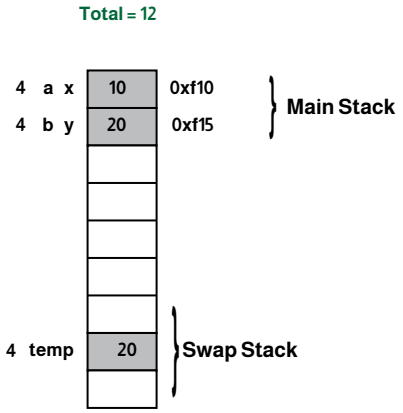
```
//V1
void Swap(int x, int y){
  int temp = x;
  x = y;
  y = temp;
}

int x = 10;
int y =20;
cout<<"X = "<<x<<endl;
cout<<"Y = "<<y<<endl;
Swap(x,y);
cout<<"X = "<<x<<endl;
cout<<"Y = "<<y<<endl;
```

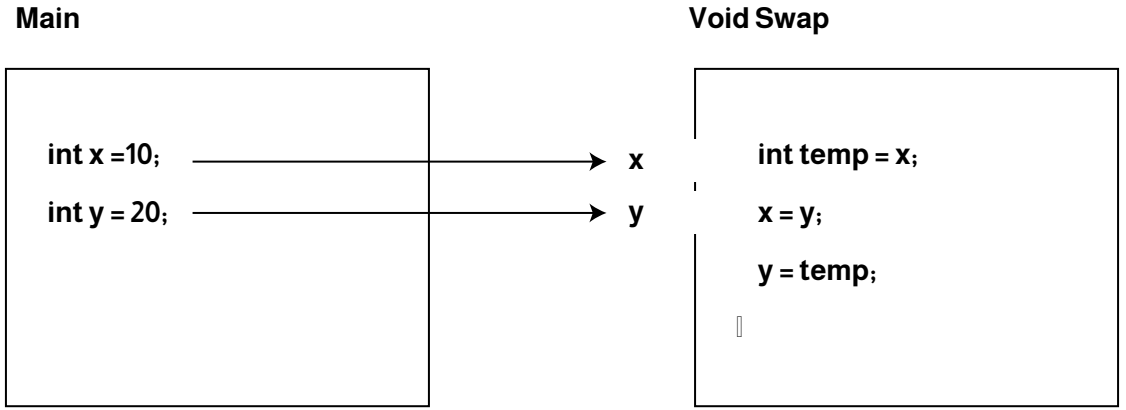



```
//V2
void Swap(int *x, int *y){
  int temp = *(x);
  *x = *y;
  *y = temp;
}

int x = 10;
int y =20;
cout<<"X = "<<x<<endl;
cout<<"Y = "<<y<<endl;
Swap(&x, &y);
cout<<"X = "<<x<<endl;
cout<<"Y = "<<y<<endl;
```



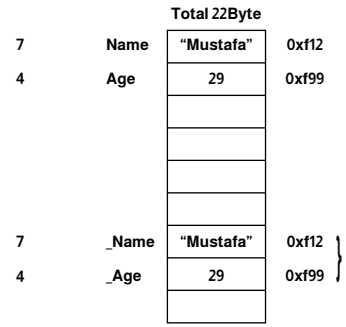

```
//V3
void Swap(int &a, int &b){
  int temp = a;
  a = b;
  b = temp;
}

int x = 10;
int y =20;
cout<<"X = "<<x<<endl;
cout<<"Y = "<<y<<endl;
Swap(x, y);
cout<<"X = "<<x<<endl;
cout<<"Y = "<<y<<endl;
```





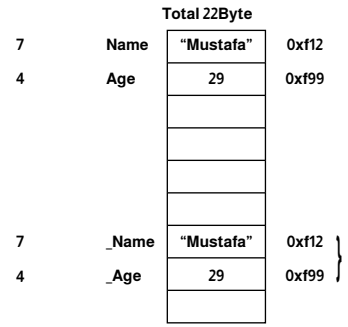
1. سئ من حيث المساحة المحجوزة
2. سئ من حيث التغير على البيانات وقت الطباعة فقط

```
1 void PrintData(string _Name, int _Age){
2     cout<<"Your Name Is "<<_Name<<endl;
3     cout<<"Your Age Is "<<_Age<<endl;
4 }
5 string Name="Mustafa";
6 int Age = 29;
7 PrintData(Name,Age);
```



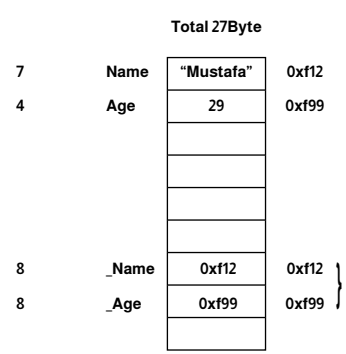
1. سئ من حيث المساحة المحجوزة
2. حافظ على البيانات وقت الطباعة دون اى تأثير

```
1 void PrintData(const string _Name,const int _Age){
2     cout<<"Your Name Is "<<_Name<<endl;
3     cout<<"Your Age Is "<<_Age<<endl;
4 }
5 string Name="Mustafa";
6 int Age = 29;
7 PrintData(Name,Age);
```



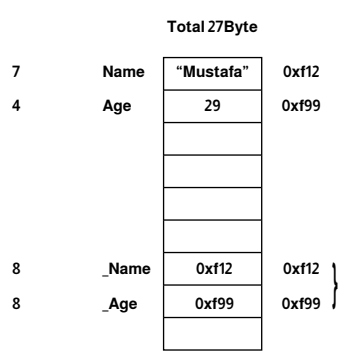
1. سئ جداً من حيث المساحة المحجوزة
2. سئ من حيث التغير على البيانات وقت الطباعة والتغير فى قيم المتغيرات الاساسية

```
1 void PrintData(string *_Name, int *_Age){
2     cout<<"Your Name Is "<<*_Name<<endl;
3     cout<<"Your Age Is "<<*_Age<<endl;
4 }
5 string Name="Mustafa";
6 int Age = 29;
7 PrintData(&Name,&Age);
```



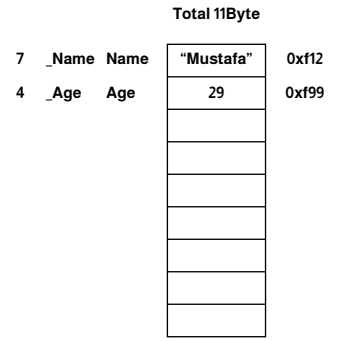
1. سئ جداً من حيث المساحة المحجوزة
2. يحافظ على البيانات من التغيير

```
1 void PrintData(const string *_Name, const int *_Age){
2     cout<<"Your Name Is "<<*_Name<<endl;
3     cout<<"Your Age Is "<<*_Age<<endl;
4 }
5 string Name="Mustafa";
6 int Age = 29;
7 PrintData(&Name,&Age);
```



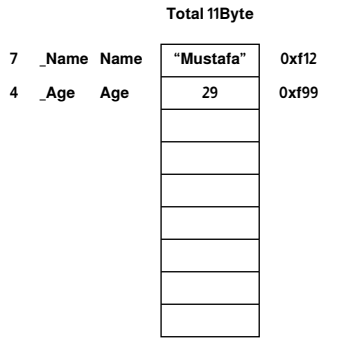
1. جيد جداً من حيث المساحة المحجوزة
2. سئ فى عدم الحفاظ على البيانات فى الطباعة و المكان الرئيسي

```
1 void PrintData(string &_Name, int &_Age){
2     cout<<"Your Name Is "<<_Name<<endl;
3     cout<<"Your Age Is "<<_Age<<endl;
4 }
5 string Name="Mustafa";
6 int Age = 29;
7 PrintData(Name,Age);
```



100%

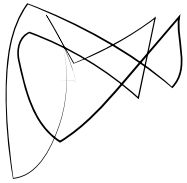
```
1 void PrintData(const string &_Name,const int &_Age){
2     cout<<"Your Name Is "<<_Name<<endl;
3     cout<<"Your Age Is "<<_Age<<endl;
4 }
5 string Name="Mustafa";
6 int Age = 29;
7 PrintData(Name,Age);
```



Best Version Of Code Of Business Logic

Function Signture

1- Function Name
2- Function Patameters
   1- Number Of Parameter
   2- Type Of Parameters



0xf99123

Function Overload

هى عملية تعريف اكتر من Function بنفس الاسم ولكن بإختلاف عدد او نوع الـParameters

```
int Sum(int x,  int y){
    return x + y;
}
int Sum(int x,  int y, int z){
    return x + y + z;
}
int Sum(int x){
    return x + 10;
}
```

```
int Sum(int x,  float y){
    return x + y;
}

int Sum(int x,  int y){
    return x + y;
}
```

Function Overload

```
int Sum(){
}

int Sum(){
}
```

Function Signture

```
int Sum(int x, int y){
  return x + y;
}

int Sum(int x, int y, int z){
  return x + y + z;
}

int Sum(int x, float y){
  return x + y;
}
int Sum(float x, int y){
  return x + y;
}
```

Default Parameters:

انك لازم تبدأ فى وضع القيم الافتراضية بداية من اليمين للشمال

Function Signture

ReturnType FunctionName (Param1 , Param2) {

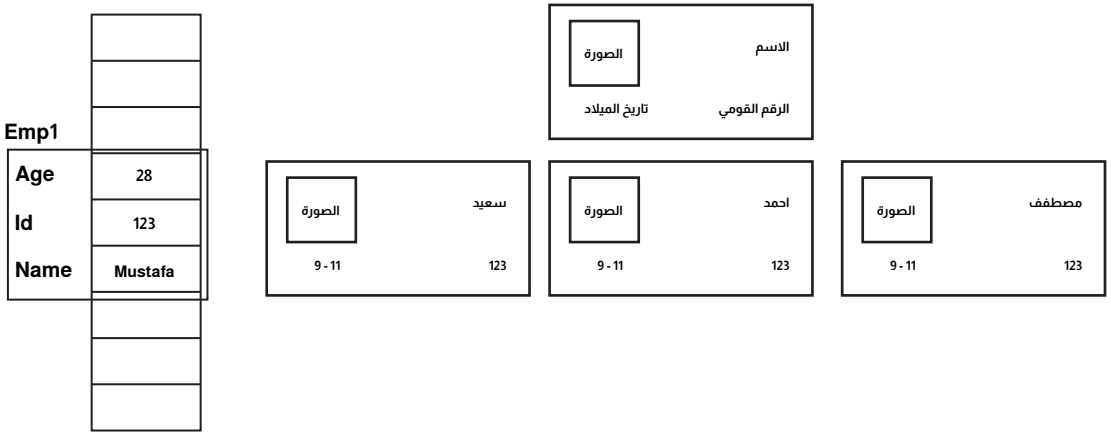
Function Body

}

Struct --> User Define Data type, Complex Data Type, Can Hold Multy Variable With Deffient Data Type

BluePrint

```
#include <iostream>

using namespace std;
struct Employee{
  int age;
  int id;
  string Name;
};
int main()
{
  Employee Emp1;
Emp1.Id = 123;
Emp1.Name="Mustafa";
Emp1.Age = 28;

  return 0;
}
```




```
#include <iostream>

using namespace std;
struct Employee{
    int age;
    int id;
    string Name;

    void Move(){
        cout<<"Employee Can Move"<<endl;
    }
    void Display(){
        cout<<"Id = "<<id<<endl;
        cout<<"Name = "<<Name<<endl;
        cout<<"Age = "<<age<<endl;
    }
};
int main()
{
    Employee Emp1;
    Emp1.id=123;
    Emp1.Name="Mustafa";
    Emp1.age = 28;

    Emp1.Display();

    return 0;
}
```

Lnaguage Lv

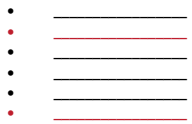
Low Lv
Mid Lv
High Lv

Execution

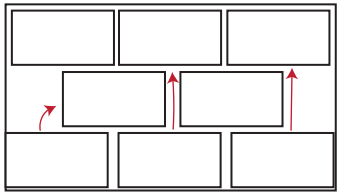
Compilation Lanuage --> EXE
Interprated Lanuage

Structre

Sequential Programming



Procedrual/ Functian Programming

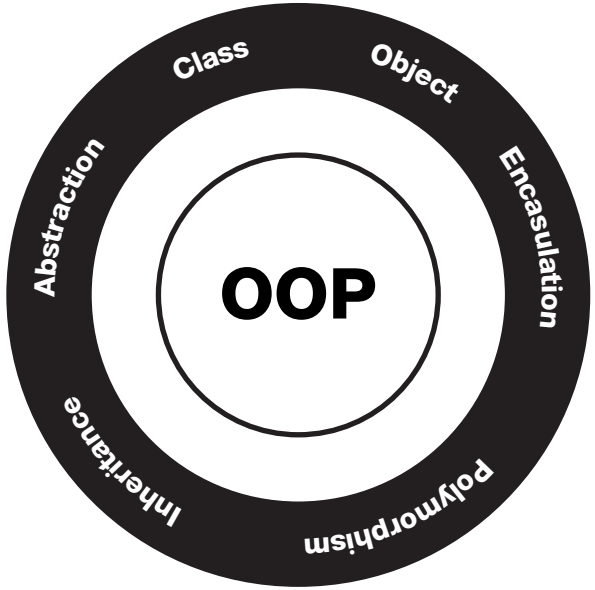


Object Oriented Programming --> Paradim

- Encapsulation
- Polymorphism
- Inheritance
- Abstract

1. CPP Is High Lv Language
2. CPP Is Compiled Lnguage
3. CPP Can Use OOP & Functional

O Object
O Oriented
P Programming




```cpp
#include <iostream>

using namespace std;
//struct Student{};
class Student{
public:
    int Grade;
    int Id;
    string Name;



};
int main()
{
    Student s1;
    s1.Grade = 100;
    s1.Id = 123;
    s1.Name="Mustafa";


    return 0;
}

```