



UNIT -IV

Object Oriented Concepts

Passing objects as Arguments

The objects of a class can be passed as arguments to member functions as well as non member functions either by value or by reference.

When an object is passed by value, a copy of the actual object is created inside the function. This copy is destroyed when the function terminates.

Passing objects as Arguments

```
class className {  
    ...  
    public:  
    void functionName(className agr1, className arg2)  
    {  
        ...  
    }  
    ...  
};  
  
int main() {  
    className o1, o2, o3;  
    o1.functionName(o2, o3);  
}
```

The diagram illustrates the process of passing objects as arguments. Two arrows originate from the `o2` and `o3` arguments in the `o1.functionName(o2, o3);` call within the `main` function. One arrow points to the `agr1` parameter in the `functionName` method signature, and the other points to the `arg2` parameter. This visualizes how the objects `o2` and `o3` are passed by reference to the function.

```

class Demo
{
    private:
        int a;
    public:
        void set(int x)
        {   a = x; }
    void sum(Demo ob1, Demo ob2)
        {a = ob1.a + ob2.a; }
        void print()
        {
            cout<<"Value of A :
"<<a<<endl;
        }
};

int main()
{
    //object declarations
    Demo d1, d2,d3;
    //assigning values to the data member of
    objects
    d1.set(10);
    d2.set(20);

    //passing object d1 and d2
    d3.sum(d1,d2);

    //printing the values
    d1.print();
    d2.print();
    d3.print();

    return 0;
}

```

Returning an Object from a function

An object can be returned by a function using the return keyword.

```
#include<iostream>

class Student {...};

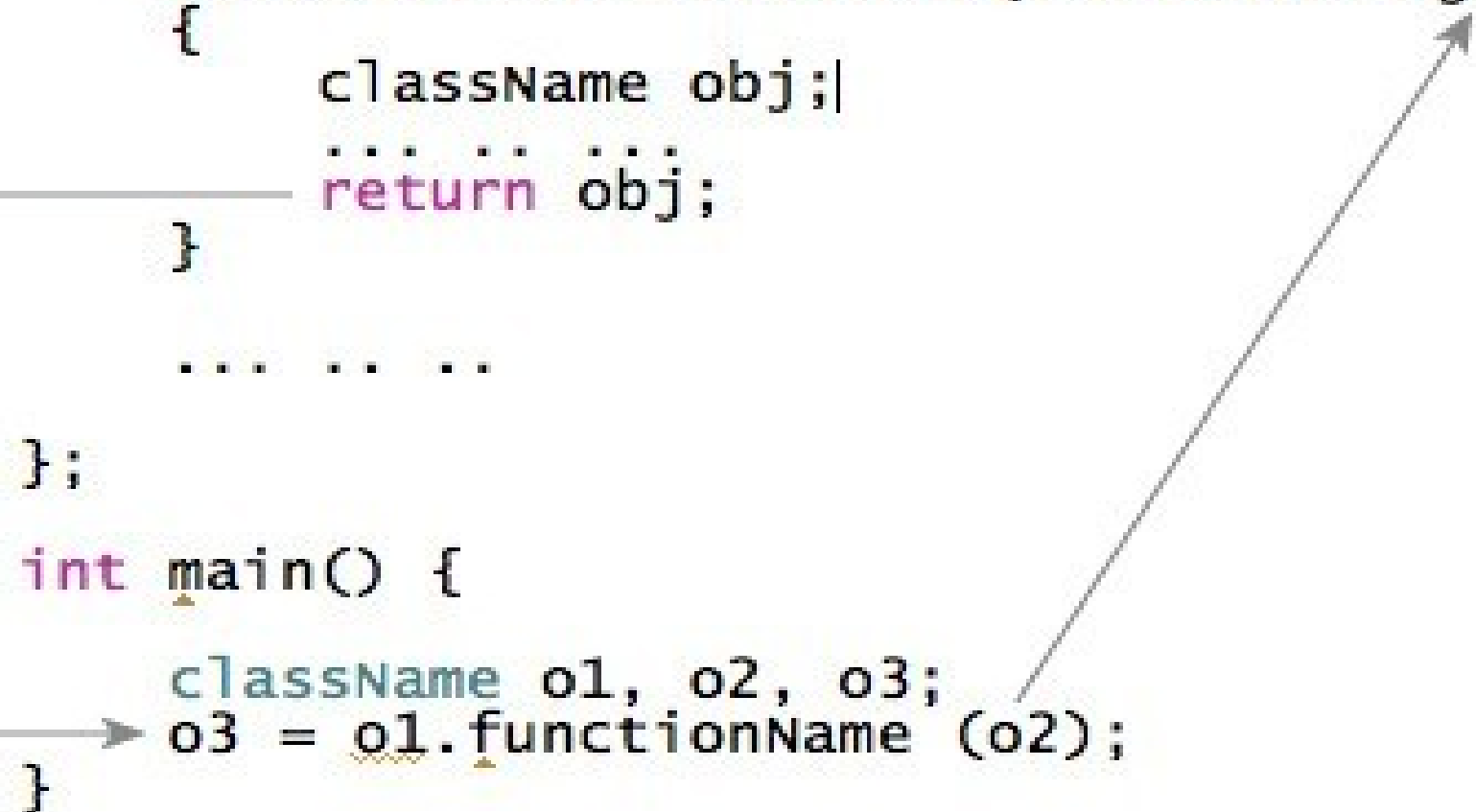
Student createStudent() {
    Student student;
    ... ..
    return student;
}

int main() {
    ... ..
    student1 = createStudent();
    ... ..
}
```

The diagram illustrates the execution flow of the provided C++ code. A teal line originates from the `createStudent()` call within the `main()` function. This line extends to the right and then turns upwards to point at the opening curly brace of the `createStudent()` function definition. A label **function call** is placed next to the vertical segment of this line. Another teal line originates from the `return student;` statement inside the `createStudent()` function and points back to the `student1 = createStudent();` assignment in the `main()` function, indicating the return of the object.

Returning an Object from a function

```
class className {  
    ... ..  
public:  
    className functionName(className agr1)  
    {  
        className obj;  
        ... ..  
        return obj;  
    }  
    ... ..  
};  
  
int main() {  
    className o1, o2, o3;  
    o3 = o1.functionName(o2);  
}
```



Arrays of Objects

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

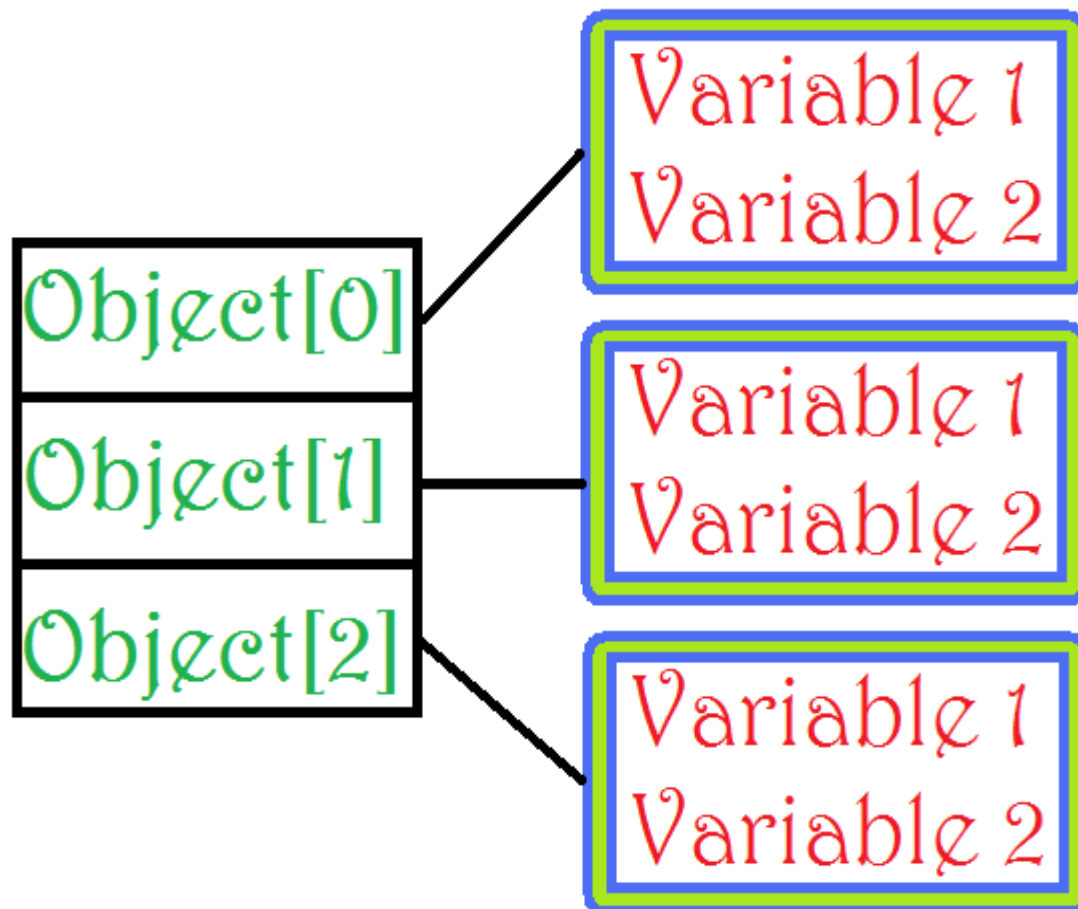
To use the data and access functions defined in the class, you need to create objects.

Syntax:

```
ClassName ObjectName[number of objects];
```

The Array of Objects stores objects. An array of a class type is also known as an array of objects.

Arrays of Objects



Arrays of Objects

```
class class-name
{
    datatype var1;
    datatype var2;
    -----
    datatype varN;

    method1();
    method2();
    -----
    methodN();
};

class-name obj[ size ];
```

```
class Employee
```

```
{  
    int id;  
    char name[30];  
    public:  
    void getdata();  
    void putdata();  
};  
void Employee::getdata()  
{  
    cout << "Enter Id : ";  
    cin >> id;  
    cout << "Enter Name : ";  
    cin >> name;  
}
```

```
void Employee::putdata()
```

```
{  
    cout << id << " ";  
    cout << name << " ";  
    cout << endl;  
}  
int main()  
{  
    Employee emp[30];  
    int n, i;  
    cout << "Enter Number of Employees  
- ";  
    cin >> n;  
    for(i = 0; i < n; i++)  
        emp[i].getdata();  
    cout << "Employee Data - " << endl;  
    for(i = 0; i < n; i++)  
        emp[i].putdata();  
}
```