1] In C++, access specifiers are keywords used in class definitions to control the visibility of class members (variables and functions) from outside the class. There are three accesss specifiers. They are- Public, Private and protected.

Example:

```
class specifiers {
public:
    int variable1;

Private:
    int variable2;
Protected:
    int variable3;
};
```

2] Default constructor:

A default constructor is a constructor that can be called without any arguments. It initializes the objects data members to default values or sets up the objects as needed.

```
class dConstructor{
public:
    int a;
    dConstructor(){
        a = 10;
        cout << "Default constructor called" << endl;
    }
};
```

Parameterized Constructor:

A parameterized constructor is a constructor that accepts parameters, allows developpers to initialize the objects data members with specific values when the object is created.

```
class pConstructor{
Public:
    int a;
    pConstructor(int variable){
        a = variable;
        cout << "Parameterized constructor called
        with a variable: " << variable << endl;
    }
};
```

Destructor:

A destructor is a special member function that gets called automatically when an object goes out of scope or is explicitly ~~let~~ deleted. It's used to release resources held by the object, perform cleanup or perform any unnecessary finalization tasks.

```
class destructor{
public:
    destructor(){
      cout << "constructor called" << endl;
    }
    ~destructor(){
      cout << "Destructor called" << endl;
    }
};
```

Here, the string "Destructor called" will automatically called.

Differences between constructors and destructors:

Constructor	Destructor
i) It initializes object state ~~~~.	i) where destructors clear up resources.
ii) They are invoked when an object is created.	ii) They are called when an object is destroyed or goes out of scope.
iii) A class can have multiple constructors	iii) A class couldn't have multiple destructors.
iv) They are called explicitly at the time of object creation using class name.	iv) They are called implicitly when the object goes out of scope or is deleted.

3/ function overloading refers to the ability to have multiple functions in the same scope with the same name but different parameter lists. The compiler determines which function to call based on the number or types of arguments passed.

Example:

```
class overloading{
Public:
    void function(int num){
        cout << "Integer number: " << num <<endl;
    }
    void function(double num){
        cout << "Double number: " << num << endl;
    }
    void function(char *c){
        cout << "character: " << c << endl;
    }
};
int main(){
    overloading obj;
    obj.function(10);
    obj.function(4.5);
    obj.display('a');}
```

4] Inheritance is a fundamental concept where a class can inherit properties and behaviors from another class. It allows the creation of a new class that is based on an existing class, acquiring its attributes and behaviors while also allowing for additional features or modifications.

Example:

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

class A {
public:
  void display(){
    cout << "Inside class A" << endl;
  }
};
class B : public A {
public:
  void show() {
    cout << "Inside class B" << endl;
};
```

```
int main(){
    B obj;
    obj.display();
    obj.show();
}
```

Advantages of Inheritance:

1. **Code Reusability:**

Inheritance promotes code reuse by allowing derived classes to inherit features from the base class, reducing redundant code.

2. **Extensibility:** It enables the creation of specialized classes (derived classes) from more general classes (base classes) by adding new features or modifying existing ones.

3. **Maintainability:** changes made to the base class can automatically reflect in the derived classes, reducing the need for modifications in multiple classes.

5] Encapsulation:

Encapsulation is a fundamental principle in object-oriented programming that involves bundling the data (attributes and properties) and the methods that manipulate the data within a single unit or class.

Abstraction:

Abstraction refers to the concept of hiding the complex implementation details and showing only the essential features of an object.

Example:

```
class Circle {
    Private:
        double radius;
    public:
        void setRadius (double r) {
            radius = r;
        }
        double Area() {
    }; return 3.1416 * radius * radius;
```

6] Advantages of using pointer:

1. Dynamic memory allocation: Pointers enable the allocation and deallocation of memory dynamically.

2. Efficient Memory management:

They allow efficient memory manipulation of memory addresses, enabling data structures, like linked lists or trees..

3. Passing by Reference:

Pointers enable passing large data structures efficiently to functions by reference rather than creating copies.

Memory Allocation using Pointer:

```
int *ptr = new int;
```

Memory Deallocation using Pointer:

```
delete ptr;
```

Example:

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

int main() {
    int *ptr = new int(10);
    cout << "Value stored at ptr: " << *ptr
                                   << endl;
    delete ptr;
    return 0;
}
```