Lecture 14

C++ Public, Protected and Private Inheritance

In C++ inheritance, we can derive a child class from the base class in different access modes. For example,

```cpp
class Base {
.... ... ....
};

class Derived : public Base {
.... ... ....
};
```

Notice the keyword `public` in the code

```cpp
class Derived : public Base
```

This means that we have created a derived class from the base class in **public mode**. Alternatively, we can also derive classes in **protected** or **private** modes.

These 3 keywords (`public`, `protected`, and `private`) are known as **access specifiers** in C++ inheritance.

public, protected and private inheritance in C++

public, **protected,** and **private** inheritance have the following features:

- **public inheritance** makes `public` members of the base class `public` in the derived class, and the `protected` members of the base class remain `protected` in the derived class.

- **protected inheritance** makes the `public` and `protected` members of the base class `protected` in the derived class.

Tasnim Niger

- **private inheritance** makes the `public` and `protected` members of the base class `private` in the derived class.

> **Note:** `private` members of the base class are inaccessible to the derived class.

```cpp
class Base {
  public:
    int x;
  protected:
    int y;
  private:
    int z;
};

class PublicDerived: public Base {
  // x is public
  // y is protected
  // z is not accessible from PublicDerived
};

class ProtectedDerived: protected Base {
  // x is protected
  // y is protected
  // z is not accessible from ProtectedDerived
};

class PrivateDerived: private Base {
  // x is private
  // y is private
  // z is not accessible from PrivateDerived
};
```

Example 1: C++ public Inheritance

```cpp
// C++ program to demonstrate the working of public inheritance

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

Tasnim Niger

```cpp
class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PublicDerived : public Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }
};

int main() {
  PublicDerived object1;
  cout << "Private = " << object1.getPVT() << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.pub << endl;
  return 0;
}
```

Output

```
Private = 1
Protected = 2
Public = 3
```

Here, we have derived `PublicDerived` from `Base` in **public mode**.

Tasnim Niger

As a result, in `PublicDerived`:

- `prot` is inherited as **protected**.
- `pub` and `getPVT()` are inherited as **public**.
- `pvt` is inaccessible since it is **private** in `Base`.

Since **private** and **protected** members are not accessible from `main()`, we need to create public functions `getPVT()` and `getProt()` to access them:

```cpp
// Error: member "Base::pvt" is inaccessible
cout << "Private = " << object1.pvt;

// Error: member "Base::prot" is inaccessible
cout << "Protected = " << object1.prot;
```

Notice that the `getPVT()` function has been defined inside `Base`. But the `getProt()` function has been defined inside `PublicDerived`.

This is because `pvt`, which is **private** in `Base`, is inaccessible to `PublicDerived`.

However, `prot` is accessible to `PublicDerived` due to public inheritance.

So, `getProt()` can access the protected variable from within `PublicDerived`.

Accessibility in public Inheritance

Accessibility	private members	protected members	public members
Base Class	Yes	Yes	Yes
Derived Class	No	Yes	Yes

Example 2: C++ protected Inheritance

```cpp
// C++ program to demonstrate the working of protected inheritance
#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;
```

```cpp
  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class ProtectedDerived : protected Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }

    // function to access public member from Base
    int getPub() {
      return pub;
    }
};

int main() {
  ProtectedDerived object1;
  cout << "Private cannot be accessed." << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.getPub() << endl;
  return 0;
}
```

Output

```
Private cannot be accessed.
Protected = 2
Public = 3
```

Tasnim Niger

Here, we have derived `ProtectedDerived` from `Base` in **protected mode**.

As a result, in `ProtectedDerived`:

- `prot`, `pub` and `getPVT()` are inherited as **protected**.
- `pvt` is inaccessible since it is **private** in `Base`.

As we know, **protected** members cannot be directly accessed from outside the class.

As a result, we cannot use `getPVT()` from `ProtectedDerived`.

That is also why we need to create the `getPub()` function in `ProtectedDerived` in order to access the `pub` variable.

```
// Error: member "Base::getPVT()" is inaccessible
cout << "Private = " << object1.getPVT();

// Error: member "Base::pub" is inaccessible
cout << "Public = " << object1.pub;
```

Accessibility in protected Inheritance

Accessibility	private members	protected members	public members
Base Class	Yes	Yes	Yes
Derived Class	No	Yes	Yes (inherited as protected variables)

Example 3: C++ private Inheritance

```cpp
// C++ program to demonstrate the working of private inheritance
#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;
```

Tasnim Niger

```cpp
  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PrivateDerived : private Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }

    // function to access private member
    int getPub() {
      return pub;
    }
};

int main() {
  PrivateDerived object1;
  cout << "Private cannot be accessed." << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.getPub() << endl;
  return 0;
}
```

Output

```
Private cannot be accessed.
Protected = 2
Public = 3
```

Here, we have derived `PrivateDerived` from `Base` in **private mode**.

As a result, in `PrivateDerived`:

- `prot`, `pub` and `getPVT()` are inherited as **private**.
- `pvt` is inaccessible since it is **private** in `Base`.

  As we know, private members cannot be directly accessed from outside the class. As a result, we cannot use `getPVT()` from `PrivateDerived`.

  That is also why we need to create the `getPub()` function in `PrivateDerived` in order to access the `pub` variable.

```
// Error: member "Base::getPVT()" is inaccessible
cout << "Private = " << object1.getPVT();

// Error: member "Base::pub" is inaccessible
cout << "Public = " << object1.pub;
```

Accessibility in private Inheritance

Accessibility	private members	protected members	public members
Base Class	Yes	Yes	Yes
Derived Class	No	Yes (inherited as private variables)	Yes (inherited as private variables)

Tasnim Niger