Object-oriented Programming

Lecture 5

CREATED BY: SYED ZAIN UL HASSAN

Constants

- The keyword ***const*** be used to declare constant variables
- They must be initialized when they are declared and cannot be modified later
- Using constant variables to specify array size makes program more *scalable*
- Constant variables are also called ***named constants*** or ***read-only variables***

Constants

```
int main()
{
    const int a = 5;
    const int b;        // will cause error
    b = 10;             // will cause error

    const int arr[] = {1, 2, 3, 4, 5};
    arr[0] = 10;        // will cause error
}
```

Principle of Least Privilege???

Principle of Least Privilege???

Limit the effect and scope of variables

Constant With Pointers

- There are four ways to use **const** with pointers:
  - **Non-constant pointers to non-constant data**
  - **Non-constant pointers to constant data**
  - **Constant pointers to non-constant data**
  - **Constant pointers to constant data**

Non-constant Pointers to Non-constant Data

- The highest access is granted by a **non-constant pointer to non-constant data**

- Data can be modified through pointer, and pointer can be made to point to other data

Non-constant Pointers With Non-constant Data

```
int main()
{
    int a = 10;
    int b = 50;

    int* pA = &a;
    *pA = 20;
    pA = &b;
}
```

Non-constant Pointers to Constant Data

- Pointer can be modified to point to any other data, but the data to which it points cannot be modified through that pointer

```
const  int * pVal;
```

Non-constant Pointers to Constant Data

```
int main()
{
    int a = 10;
    int b = 50;

    const int* pA = &a;
    *pA = 20;          // this line will cause error
    pA = &b;
}
```

Constant Pointers to Non-constant Data

- Always points to the same memory location, but the data at that location can be modified through the pointer

```
int * const pVal = &val;
```

Constant Pointers to Non-constant Data

```
int main()
{
    int a = 10;
    int b = 50;

    int* const pA = &a;
    *pA = 20;
    pA = &b;          // this line will cause error
}
```

Constant Pointers to Constant Data

- Always points to the same memory location, and the data at that location cannot be modified via the pointer

```
const int * const pVal = &val;
```

Constant Pointers to Constant Data

```
int main()
{
    int a = 10;
    int b = 50;

    const int* const pA = &a;
    *pA = 20;          // cannot do this
    pA = &b;           // cannot do this as well
}
```

Constant Parameter

When a function parameter is declared as a constant, it cannot be modified inside that function

```
void func(const int a, int b)
{
    a += 2;        // this line will give an error
    b += 3;
    cout << a;
}
```

Constant Function

When const keyword is used as following, the function cannot make changes to any member variables of the class

```
class A
{
        int x;

        public:
        void func() const
        {
                x = 10;          // this line will give an error
        }
        // other code
};
```

Constant Function

```
class A
{
        int x;

        public:
        void func(int a) const
        {
                a = 5;          // no problem here
                x = 10;         // this line will give an error
        }
        // some other code
};
```

Constant Object

When an object is declared as constant, it can only call constant functions i.e. such an object cannot, thus, make any changes to member variables

const MyClass *ob_name*;