Object Oriented Programming

Classes: `static` keyword

Mr. Usman Wajid

usman.wajid@nu.edu.pk

March 9, 2023

National University
of Computer & Emerging Sciences

a `struct` vs a `class`

- By default, members of a `struct` are `public`
  - However, a member can be made private by using the `private` specifier in `struct`

a `struct` vs a `class`

- By default, members of a `struct` are `public`
  - However, a member can be made private by using the `private` specifier in `struct`
- By default, the members of a `class` are `private`

a `struct` vs a `class`

- By default, members of a `struct` are `public`
  - However, a member can be made private by using the `private` specifier in `struct`
- By default, the members of a `class` are `private`
- in C++, the capability of `struct` is expanded to include member function, constructors and destructors

a `struct` vs a `class`

- By default, members of a `struct` are `public`
  - However, a member can be made private by using the `private` specifier in `struct`
- By default, the members of a `class` are `private`
- in C++, the capability of `struct` is expanded to include member function, constructors and destructors
- `classes` and `struct` have the same capabilities

a `struct` vs a `class`

- By default, members of a `struct` are `public`
  - However, a member can be made private by using the `private` specifier in `struct`
- By default, the members of a `class` are `private`
- in C++, the capability of `struct` is expanded to include member function, constructors and destructors
- `classes` and `struct` have the same capabilities
- if all member variables of a `class` are `public` then use `struct` instead

`static` keyword

`static` keyword

A static member variables of a class exist even if no object of that `class` exist. Similarly, if an object of a class is destroyed, the `static` variables of that class still resides in the memory.

- The `static` keyword is used to declare a function or variable of a `class` as `static`

`static` keyword

`static` keyword

A static member variables of a class exist even if no object of that `class` exist. Similarly, if an object of a class is destroyed, the `static` variables of that class still resides in the memory.

- The `static` keyword is used to declare a function or variable of a `class` as `static`
- A `public static` function or variable or a class can be accessed using the class name and the scope resolution (::) operator

`static` keyword

`static` keyword

A static member variables of a class exist even if no object of that `class` exist. Similarly, if an object of a class is destroyed, the `static` variables of that class still resides in the memory.

- The `static` keyword is used to declare a function or variable of a `class` as `static`
- A `public static` function or variable or a class can be accessed using the class name and the scope resolution (::) operator
- A `static` member function of a class can only access a `static` member variable

`static` keyword

`static` keyword

A static member variables of a class exist even if no object of that `class` exist. Similarly, if an object of a class is destroyed, the `static` variables of that class still resides in the memory.

- The `static` keyword is used to declare a function or variable of a `class` as `static`
- A `public static` function or variable or a class can be accessed using the class name and the scope resolution (::) operator
- A `static` member function of a class can only access a `static` member variable
- `static` member variable should be initialized outside the class body by using the class name and the scope resolution operator (::)

`static` keyword

`static` keyword

A static member variables of a class exist even if no object of that `class` exist. Similarly, if an object of a class is destroyed, the `static` variables of that class still resides in the memory.

- The `static` keyword is used to declare a function or variable of a `class` as `static`
- A `public static` function or variable or a class can be accessed using the class name and the scope resolution (::) operator
- A `static` member function of a class can only access a `static` member variable
- `static` member variable should be initialized outside the class body by using the class name and the scope resolution operator (::)
- `static const` member variable should be initialized within the class

`class static const` variable initialization

- the static const variable has to be declared and initialized at the same time within the `class` body

```
class Student {

        private:
                static const int max_students = 250 ;

};
```

`class static` variable initialization

- the `static` variable must be defined inside the `class` body
- a `static` variable is initialized outside the `class` body by using the class-name and the scope resolution operator (::)

```
class Student {

        private:
                static int std_count;

};

int Student::std_count=0;
```

`class static` variable Example 1

```
class Student {
        private:
        static int std_count;
        public:
        Student(){
                std_count++;
        }
        void set_std_count(int std_count){
                this->std_count = std_count;
        }
        int get_std_count(){
                return std_count;
        }
};
int Student::std_count=0;
int main() {
        Student ali, mahad, zain;
        cout<<"total students: "<<zain.get_std_count()<<
            endl;
}
```

```
total students: 0
```

`class static` variable Example 2

```
class Student {
        public:
                static int std_count;
                Student(){
                        std_count++;
                }
};

int Student::std_count=0;

int main() {

        cout<<"total students: "<<Student::std_count<<endl;

        Student ali, mahad, zain;

        cout<<"total students: "<<Student::std_count<<endl;
}
```

```
total students: 0
total students: 3
```

`class static` variable Example 3

```
class Student {
        public:
                static int std_count;
                Student(){
                        std_count++;
                }
                ~Student(){
                        std_count--;
                }
};
int Student::std_count=0;
int main() {
        cout<<"total students: "<<Student::std_count<<endl;
        Student * x = new Student;
        cout<<"total students: "<<Student::std_count<<endl;
        Student ali, mahad, zain;
        cout<<"total students: "<<Student::std_count<<endl;
        delete x;
        cout<<"total students: "<<Student::std_count<<endl;
}
```

```
total students: 0
total students: 1
total students: 4
total students: 3
```