DEFINING CONSTANTS

C++ BASICS

prepared by:

Gyro A. Madrona
Electronics Engineer

TOPIC OUTLINE

`conts` Keyword

Preprocessor Macros

Enumerations

DEFINING CONSTANTS

CONSTANTS

Constants are values that **cannot be changed** during the execution of a program. They improve code readability, maintainability, and safety by preventing accidental modifications to critical values.

CONST KEYWORD

The **const** keyword is used to declare a variable as constant. Once initialized, the value of a **const** variable cannot be modified.

Example:

```
int main()
{
    double const PI = 3.14;
    PI = 4.0;
    return 0;
}
```

```
error: assignment of read-only variable
'PI'.
```

PREPROCESSOR MACROS

The **#define** directive is a preprocessor macro used to define constants. It is not type-safe and is replaced by the preprocessor before compilation.

Example:

```
#include <iostream>
#define PI 3.14
using namespace std;
int main()
{
    double r = 1.0;
    double c = 2*PI*r;
    return 0;
}
```

ENUMARATIONS

An enumeration (or **enum**) is a user-defined type that consists of a **set of named integer constants**. By default, the first enumerator is assigned the value **0**.

Example:

```
int main()
{
    enum Days {Sunday,Monday,Tuesday};
    cout << Monday;
    return 0;
}
```

ENUMARATIONS

You can explicitly assign integer values to enumerators.

Example:

```
int main()
{
    enum Price {
        Apple = 20,
        Orange = 15,
        Grape = 35
    };
    cout << Orange;
    return 0;
}
```

EXERCISE

Determine the output of this code:

```
#include <iostream>
using namespace std;
int main()
{
    enum Color {Black,Brown,Red};
    Color code = Red;
    cout << code;
    return 0;
}
```

Determine the output of this code:

```
#include <iostream>
using namespace std;
int main()
{
    enum Color {
       Brown = 1,
       Orange = 3,
       Blue = 6,
    }code;
    code = Blue;
    cout << code;
    return 0;
}
```

LABORATORY