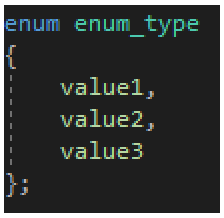
Object Oriented programming and software engineering – Lab 4

Adam Korytowski - 2025

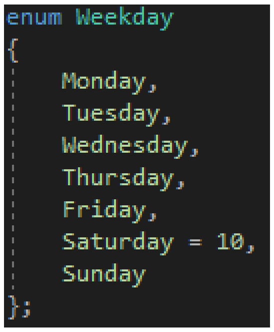
1. Enumeration

Enumeration is user defined int type that consists internal constant and each integral constant is given a name (such constant can be called identifier). To define enumerated integer data type, function enum is used.



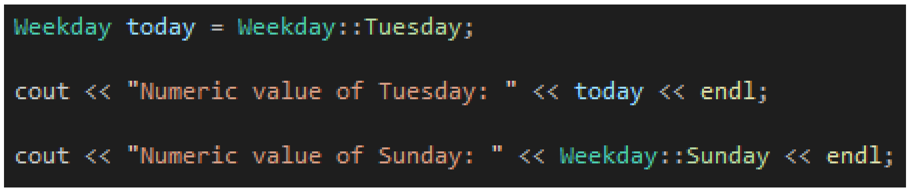
```
enum enum_type
{
    value1,
    value2,
    value3
};
```

enum_type is the name of enumerated data type. value1, value2,... are values of type name. By default, value1 will be equal to 0, value2 will be 1 and so on but, the programmer can change the default value as seen in the example (Saturday=10).



```
enum Weekday
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday = 10,
    Sunday
};
```

Example:



```
Weekday today = Weekday::Tuesday;

cout << "Numeric value of Tuesday: " << today << endl;

cout << "Numeric value of Sunday: " << Weekday::Sunday << endl;
```

Task: Enum for Traffic Lights (2 pts)

Objective:

Create an enum to represent traffic light colors and use it in a function to decide what action a driver should take.

Instructions:

- Define an enum called TrafficLight with values Red, Yellow, Green.
- Write a function getAction() that takes a TrafficLight value and returns a corresponding action ("Stop", "Wait", "Go").
- In main(), ask the user to input a number (0 for Red, 1 for Yellow, 2 for Green).
- Convert the input into an enum and call getAction() to display the action.

2. Typedef

typedef is a keyword in C++ that allows you to create an alias (alternative name) for an existing data type. This makes code easier to read, write, and maintain, especially for complex types like pointers and structures.

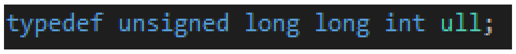
Key Features of typedef:

- Simplifies long or complex type definitions.
- Makes code more readable and understandable.
- Can be used with basic types, structs, pointers, and more.

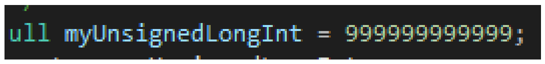
Applications of typedef

- Making complex types easier to use (e.g., unsigned long long int → ull).
- Improving code readability and portability.
- Creating shorter names for structures and pointers.
- Defining function pointer types

Example:



```cpp
typedef unsigned long long int ull;
```



```cpp
ull myUnsignedLongInt = 999999999999;
```

Task: Typedef for a Pointer Type (2 pts)

Objective:

Use typedef to simplify the declaration of a pointer to an integer.

Instructions:

1. Use typedef to define IntPtr as an alias for int*.
2. Dynamically allocate an integer using new and assign it to an IntPtr variable.
3. Modify and print the value of the allocated memory.
4. Deallocate the memory using delete.

3. Function overloading

Function overloading in C++ allows multiple functions with the same name but different parameters (type, number, or both). The compiler determines which function to call based on the arguments passed.

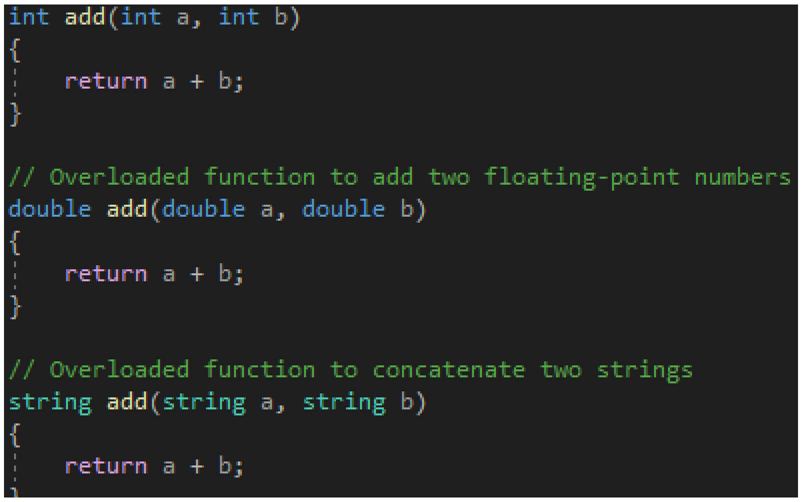
Key Features of Function Overloading:

- Improves code readability by using the same function name for related operations.
- Allows different implementations of a function for different data types.
- The function signature (parameters) must be unique for each overloaded function.

Applications of Function Overloading

- Performing similar operations on different data types (e.g., adding integers, floating points).
- Handling different numbers of parameters (e.g., a print() function that prints one or multiple values).
- Providing default behavior while allowing customization (e.g., a function that works with or without optional arguments).

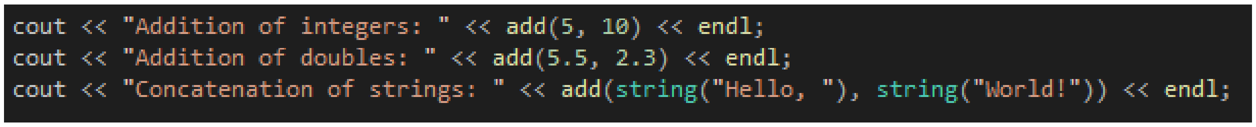
Example:



```
int add(int a, int b)
{
    return a + b;
}

// Overloaded function to add two floating-point numbers
double add(double a, double b)
{
    return a + b;
}

// Overloaded function to concatenate two strings
string add(string a, string b)
{
    return a + b;
```



```
cout << "Addition of integers: " << add(5, 10) << endl;
cout << "Addition of doubles: " << add(5.5, 2.3) << endl;
cout << "Concatenation of strings: " << add(string("Hello, "), string("World!")) << endl;
```

Task: Overloaded Function for Area Calculation (2 pts)

Objective:

Create an overloaded function area() that calculates the area of a square, rectangle, and circle.

Instructions:

- Overload area() to accept:
    - One integer (side of a square).
    - Two integers (length and width of a rectangle).
    - One double (radius of a circle, assume π = 3.1416).
- Call all versions of area() in main() and print the results.