

# Computer Programming

Dr. Deepak B Phatak  
Dr. Supratik Chakraborty  
Department of Computer Science and Engineering  
IIT Bombay

Session: Names and Type Declarations in C++

# Quick Recap of Some Relevant Topics

---



- Structure of a simple C++ program
- “main” function
- Variables and declarations

# Overview of This Lecture

---



- Names in C++
- Type declarations
- Examples: variables, functions

# Names in C++

---



- In C++, names are used to represent ‘objects’
- Each object can have a value of certain type
  - Thus every name must have an associated type
- Values associated with named objects can change as program executes
- Constant values used in the program are also objects of a certain type

# Names in C++

---

- Name is any sequence of characters from A to Z, a to z, 0 to 9 and underscore (“\_”)
  - **Cannot** start with 0-9: **1MyVar** not ok
  - **Cannot** be a C++ keyword: namespace, int, return, ...
  - **Can** start with \_ or any letter from A to Z, a to z: **\_MyVar\_1** ok
  - **Can** be any length
    - Some compilers may limit length to some large number
    - Not a real concern in practice
- Meaningful names important for readability of program
  - Variable named **averageMarks** says what it is used for
  - Variable named **xyz** makes it difficult to understand its purpose

# Type Declarations in C++

---

- C++ allows several types of values  
`char, int, float, double, void` (valueless), `bool` ...
- Type of object must be declared before its use
  - Format: `typeName` `objectName`
  - Example: `int midSemMarks;`  
`char yesNoResponse;`
- Compiler uses declarations to allocate memory space based on type
  - Example: `midSemMarks` requires 4 bytes  
`yesNoResponse` requires 1 byte

# Type Declarations in C++

---



- There exist ‘qualifiers’ to certain types
  - short int (2 bytes)
  - unsigned int (no sign bit)
  - long int (4 or 8 bytes)
  - unsigned char (1 byte for storing unsigned integer)
- Compiler uses qualifier to decide how much space to allocate and how to interpret stored value

# Simple C++ Functions and Types

---



- Encapsulate a computational (sub)-task and give it a name
  - Same naming rules as for variables and objects
- Can accept optional input parameters and return values
  - Input parameters have names and types
  - Return value has type
  - Parameterized computation

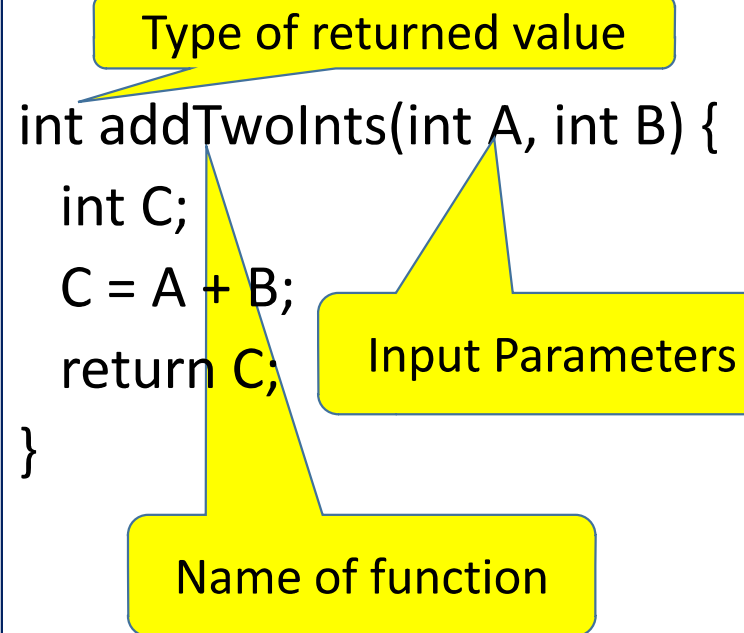


# Simple C++ Functions and Types

## Our friendly summing program:

```
int main() {  
    int A, B, C;  
    cout << "Give two numbers";  
    cin >> A >> B;  
    C = A + B;  
    cout << "Sum is" << C;  
    return 0;  
}
```

## Function to add integers:



The diagram shows the function definition `int addTwoInts(int A, int B) { ... }` with three yellow callout boxes. One box labeled 'Type of returned value' points to the `int` at the start of the function signature. Another box labeled 'Input Parameters' points to the `int A, int B` part. A third box labeled 'Name of function' points to `addTwoInts`.

```
int addTwoInts(int A, int B) {  
    int C;  
    C = A + B;  
    return C;  
}
```

# Simple C++ Functions and Types

- Just like variables, name and type of function must be declared before its use

```
int addTwoInts(int A, int B);
```

```
int addTwoInts(int A, int B) {  
    int C;  
    C = A + B;  
    return C;  
}
```

```
int main() {  
    int A, B, C;  
    cout << "Give two numbers";  
    cin >> A >> B;  
    C = addTwoInts(A, B);  
    cout << "Sum is" << C;  
    return 0;  
}
```

# Summary

---



- Names in C++
  - Variables, functions ...
- Type declarations
  - Some commonly used types