Variables in C++:

What is a Variable?

A variable in C++ is a named storage location in memory that holds a value. It acts as a container to store data that can be used and modified during program execution.

Rules for Naming Variables

1. The name must start with a letter or an underscore (_).
2. The rest of the name can include letters, digits, and underscores.
3. No spaces or special characters are allowed (e.g., @, #, !).
4. Cannot use C++ keywords (e.g., int, return, for).
5. Variable names are case-sensitive (age and Age are different).

Types of Variables

1. Integer (int)

- Stores whole numbers without decimals.
- Range depends on the system (e.g., typically -2,147,483,648 to 2,147,483,647).

Example:

```
#include <iostream>

using namespace std;

int main() {

  int age = 25; // Declare and initialize an integer variable

  cout << "Age: " << age << endl;

  return 0;

}
```

Output:

```
Age: 25
```

---

2. Floating-Point (float and double)

- **float**: Stores single-precision decimal numbers (less precision).
- **double**: Stores double-precision decimal numbers (more precision).

Example:

```
#include <iostream>
using namespace std;
int main() {
   float pi = 3.14f; // 'f' denotes a float
   double e = 2.718281828459; // Double for higher precision
   cout << "Pi: " << pi << endl;
   cout << "Euler's number: " << e << endl;
   return 0;
}
```

Output:

```
Pi: 3.14
Euler's number: 2.71828
```

---

3. Character (char)

- Stores a single character using single quotes (').
- Requires 1 byte of memory.

Example:

```
#include <iostream>
using namespace std;
int main() {
   char grade = 'A'; // Declare a character variable
   cout << "Grade: " << grade << endl;
   return 0;
}
```

Output:

```
Grade: A
```

---

4. String (string)

- Stores sequences of characters.
- Requires the <string> library.

Example:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
   string name = "John Doe"; // Declare a string variable
   cout << "Name: " << name << endl;
   return 0;
}
```

Output:

```
Name: John Doe
```

---

5. Boolean (bool)

- Stores true (1) or false (0).

Example:

```
#include <iostream>
using namespace std;
int main() {
   bool isAdult = true; // Boolean variable
   cout << "Is adult: " << isAdult << endl;
   return 0;
}
```

Output:

```
Is adult: 1
```

---

6. Constant (const)

- Variables declared as const cannot be changed after initialization.

Example:

```
#include <iostream>
using namespace std;
int main() {
    const double PI = 3.14159; // Declare a constant
    cout << "Pi: " << PI << endl;
    // PI = 3.14; // This would cause a compile-time error
    return 0;
}
```

Output:

```
Pi: 3.14159
```

Variable Declaration and Initialization:

Declaration

- Tells the compiler to allocate memory for the variable.
- Syntax:

```
data_type variable_name;
```

Initialization

- Assigns a value to the variable at the time of declaration.
- Syntax:

```
data_type variable_name = value;
```

Example:

```
int age;       // Declaration
age = 25;      // Initialization
int height = 170; // Declaration and Initialization
```

Scope of Variables

1. **Local Variables**:
   - Declared inside a function or block.
   - Accessible only within the function/block.

Example:

```
#include <iostream>
using namespace std;
int main() {
    int x = 10; // Local variable
    cout << "x: " << x << endl;
    return 0;
}
```

2. **Global Variables**:
   - Declared outside any function.
   - Accessible by all functions in the program.

Example:

```
#include <iostream>
using namespace std;
int x = 10; // Global variable
int main() {
    cout << "x: " << x << endl;
    return 0;
}
```

3. **Static Variables**:
   - Retain their value between function calls.

Example:

```
#include <iostream>
using namespace std;
void counter() {
    static int count = 0; // Static variable
    count++;
    cout << "Count: " << count << endl;
}

int main() {
    counter();
    counter();
    counter();
    return 0;
}
```

Output:

```
Count: 1
Count: 2
Count: 3
```

Input and Output with Variables:

Variables are often used with cin and cout for interaction with the user.

Example:

```cpp
#include <iostream>
using namespace std;
int main() {
    string name;
    int age;
    cout << "Enter your name: ";
    cin >> name;
    cout << "Enter your age: ";
    cin >> age;
    cout << "Hello, " << name << "! You are " << age << " years old." << endl;
    return 0;
}
```

Input:

```
Enter your name: Alice
Enter your age: 25
```

Output:

```
Hello, Alice! You are 25 years old.
```

Key Points

1. Use descriptive names for variables (e.g., age, score).
2. Choose the correct data type for the data you want to store.
3. Always initialize variables before use.
4. Understand the scope of variables to avoid unexpected behavior.