

Tishk International University
Cybersecurity Department
Course Code: CBS 113



Programming Fundamentals

Lecture 2



Variables

Fall 2025

Hemin Ibrahim, PhD
hemin.ibrahim@tiu.edu.iq

Outline



- C++ Variables and types
- Declaration and Initialization
- Data type range
- Overflow and underflow
- Identifiers
- Variable scope
- Constants
- cin Object

Objectives



- Understand the Concept of Variables
- Learn about fundamental data types in C++.
- Importance of declaring variables and initializing them with values
- Learn how to define and use constants in C++ for immutable values
- Understand the cin object for interactive input from the standard input stream.

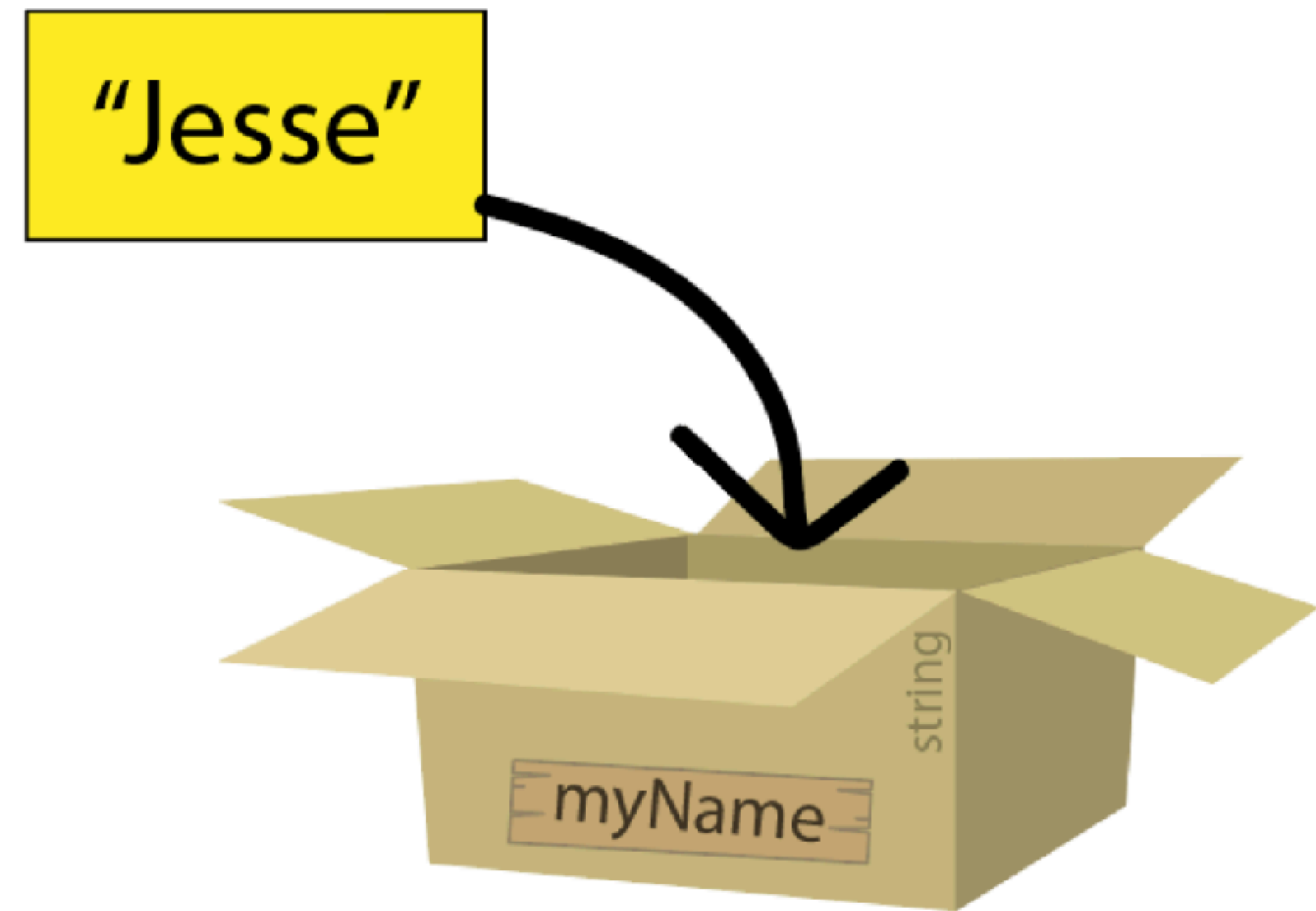
- A named storage location in computer memory
- Like a labeled container that holds a specific type of data
- Has a name, type, value, and memory address

$$2x + 1 = 9$$

variable

$$A = 4 \times S$$

variables

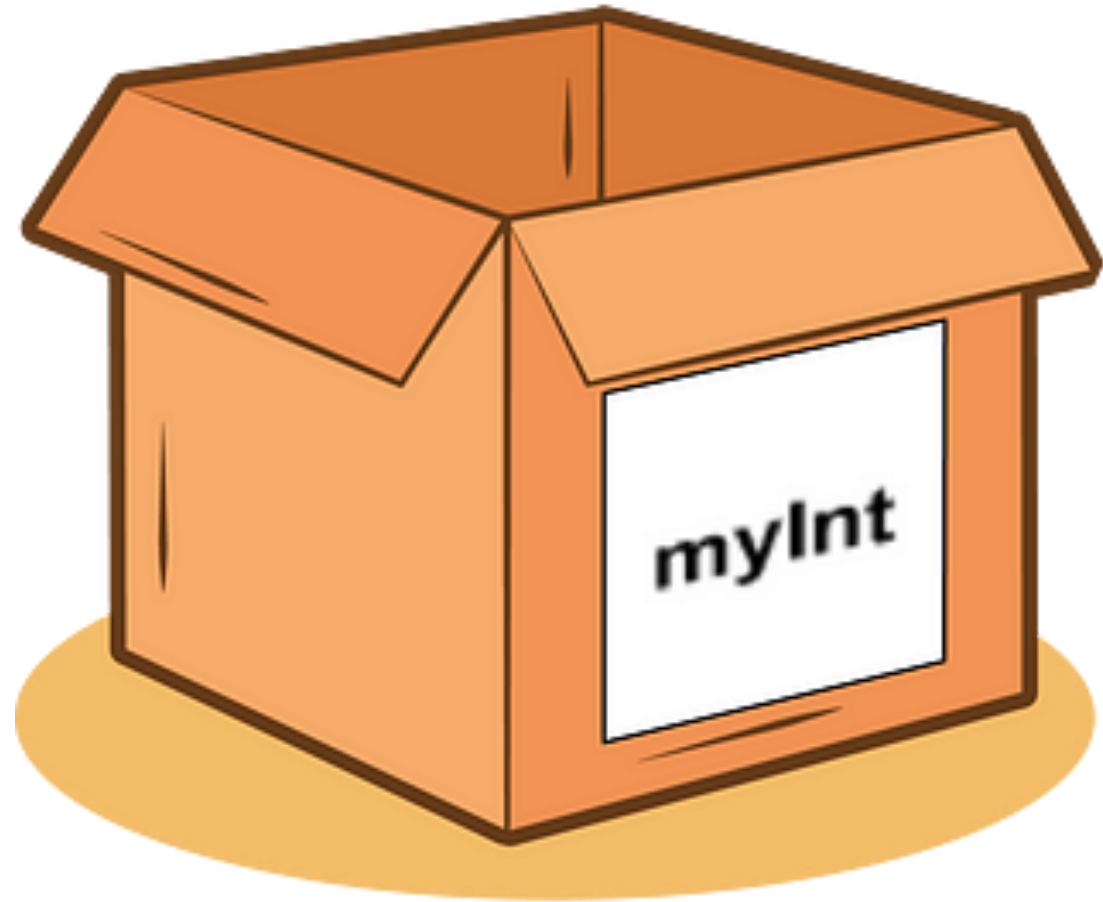


Variables are containers for storing data values.

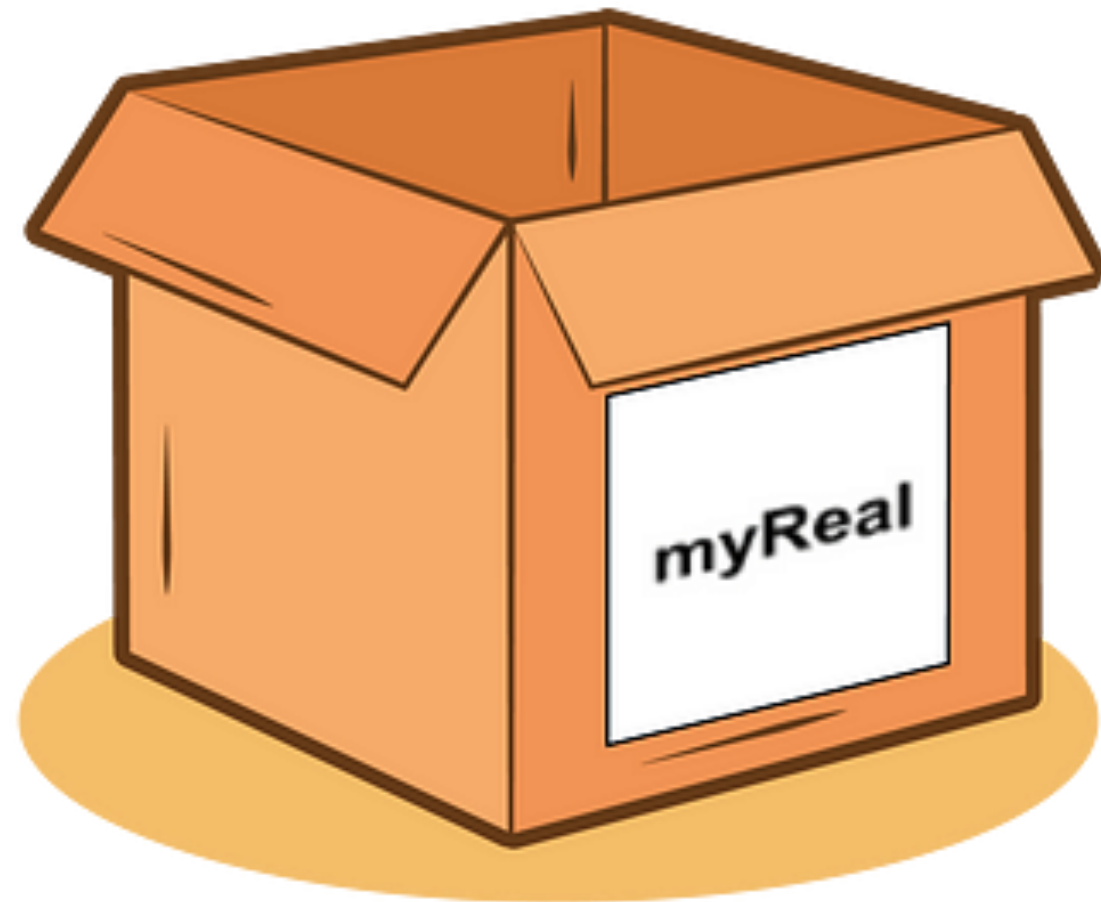
In C++, there are different types of variables (defined with different keywords), for example:

- **int** - stores integers (whole numbers), without decimals, such as 123 or -123
- **double, float** - stores floating point numbers, with decimals, such as 19.99 or -19.99
- **char** - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- **string** - stores text, such as "Hello World". String values are surrounded by double quotes
- **bool** - stores values with two states: true or false

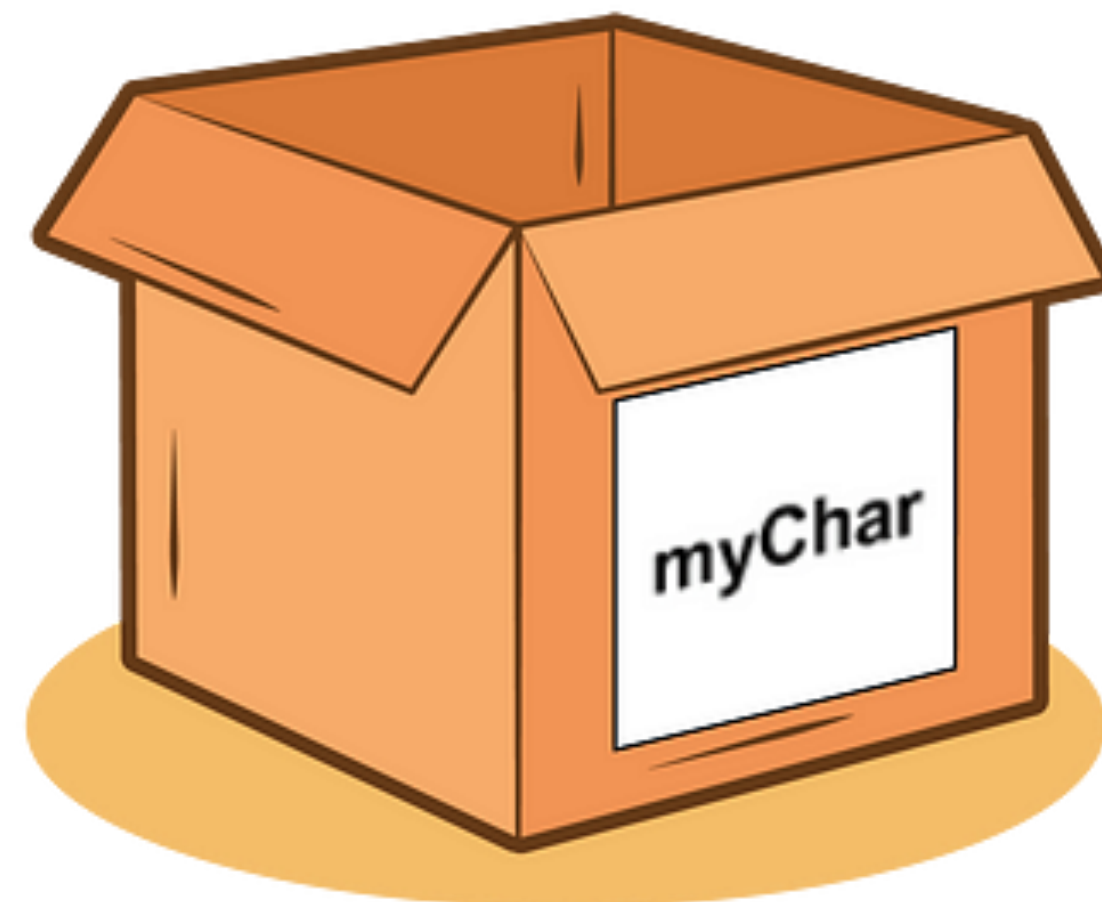
4



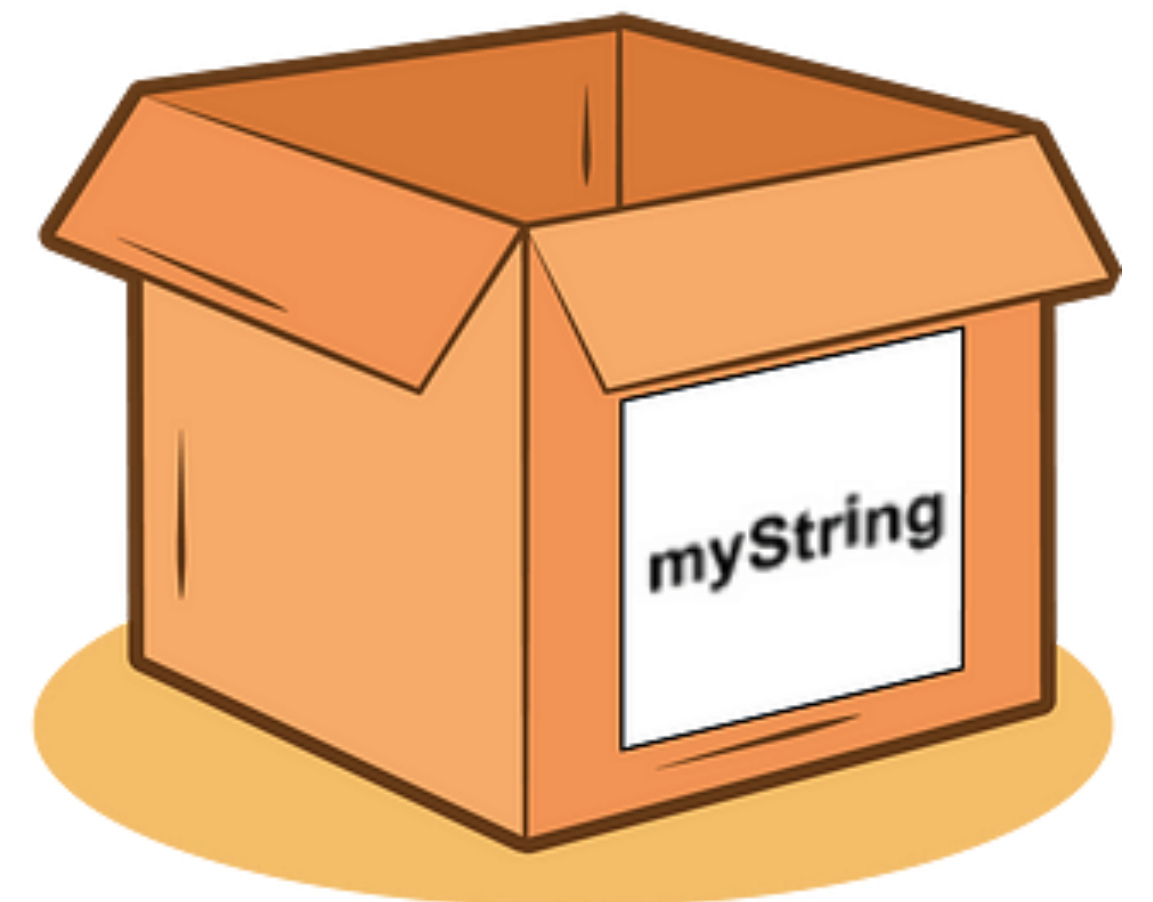
2.5



'a'



"hello"



10 minutes activity



- Look around your daily life
- Match real-world examples to data types
- Share your best examples with the class

Think about

- Your student ID card
- Temperature readings
- Traffic lights
- Your full name
- Your age

int - Number of students in class

double - Your **GPA** (3.85)

char - Your grade in last **exam** ('A')

string - Your email address

bool - Is it raining? (**true/false**)

Declaration with initialization

data_type variable_name = value; // Declaration and initialization

data_type variable_name; // Declaration only

variable_name = value; // Assignment (initialization)

C++ Variables - int



The **int** data type is used for storing whole numbers without decimals. It's suitable for representing quantities that are counted in whole units.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      int numberOfStudents = 30;
6
7      int x;
8      x = 7;
9
10     return 0;
11 }
```

Type of the variable

Variable identifier

The **int** data type is used for storing whole numbers without decimals. It's suitable for representing quantities that are counted in whole units.

```
int age = 20;
```

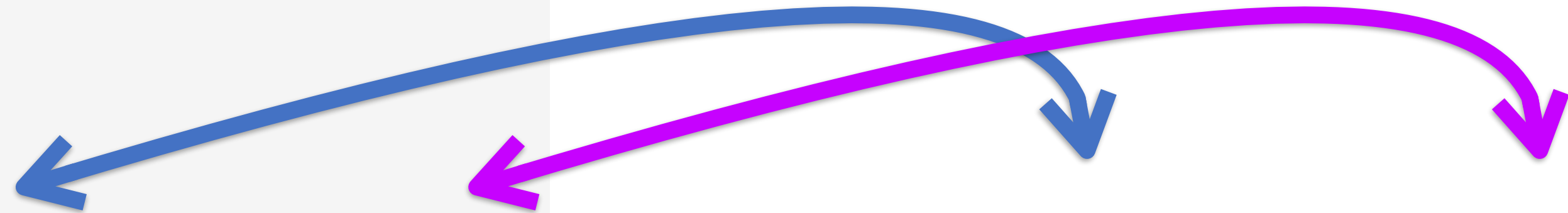
Diagram illustrating the components of the C++ code snippet `int age = 20;`:

- int**: Data Type
- age**: Variable_name
- 20**: Value

Declaration and Initialization



```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      int numberOfStudents = 30;
6
7      int x;
8      x = 7;
9
10     return 0;
11 }
```



← Declaration and Initialization together

← Declaring (Creating) Variable

← Initializations of a Variable

```
int score1 = 95, score2 = 87, score3;  
score3 = 56;
```

C++ Variables - Floating-point



The **double** and **float** data types are used for storing numbers with decimals. **double** provides higher precision compared to **float**.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      double todayTemperature = 18.4;
6      float price = 99.9;
7
8      double quiz1;
9      quiz1 = 3.4;
10
11     return 0;
12 }
```

Example:

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      cout << 3.1415 << endl;    // floating-point = 3.1415
6      cout << .1415 << endl;    // floating-point = 0.1415
7      cout << 3.0 << endl;      // floating-point = 3
8      cout << 3. << endl;       // floating-point = 3
9      cout << 31415E-4 << endl;  // floating-point = 3.1415
10     cout << 31415E-4L << endl; // floating-point = 3.1415 long double
11     cout << 31E2 << endl;      // floating-point = 3100
12
13
14     return 0;
15 }
```

C++ Variables - Character (char)



The **char** data type is used for storing single characters. Characters are enclosed in single quotes.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      char studentGrade = 'A';
6      char x = '@';
7      char num = '1';
8
9      return 0;
10 }
```

The **string** data type is used for storing sequences of characters (strings). Strings are enclosed in double quotes.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      string courseName = "Programming 1";
6      string fullName;
7      fullName = "Kurda Balen Ahmed";
8
9      return 0;
10 }
```

The **bool** data type is used for storing values with two states: **true** or **false**. It's often used in decision-making and control flow.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      bool isTestPassed = true;
6      bool evenNumber = false;
7
8      return 0;
9  }
```

Activity #1



Find errors in this code

```
int decimal = 3.14;  
char fullName = "John";  
bool score = 75;  
float preciseCalculation = 123456789.123456789;
```

Type Selection

- Storing phone numbers
- Calculating price with tax
- Tracking exam scores
- Recording student attendance
- Storing Book Titles
- Measuring Temperature
- Counting Pages in a Book
- Determining Pass/Fail Status

Data types range



Data Type	Size (bytes)	Range
<code>`bool`</code>	1	<code>`true`</code> or <code>`false`</code>
<code>`char`</code>	1	-128 to 127 (signed), 0 to 255 (unsigned)
<code>`short`</code>	At least 2	-32768 to 32767
<code>`int`</code>	At least 4	-2147483648 to 2147483647
<code>`long`</code>	At least 4	-2147483648 to 2147483647
<code>`long long`</code>	At least 8	-9223372036854775808 to 9223372036854775807
<code>`float`</code>	4	Approximately $\pm 3.4e38$ (7 decimal digits)
<code>`double`</code>	8	Approximately $\pm 1.7e308$ (15 decimal digits)
<code>`long double`</code>	At least 10	Platform-dependent, greater precision than <code>`double`</code>

Real-World Examples:??

Data Type	Size (bytes)	Range
<code>`bool`</code>	1	<code>`true`</code> or <code>`false`</code>
<code>`char`</code>	1	-128 to 127 (signed), 0 to 255 (unsigned)
<code>`short`</code>	At least 2	-32768 to 32767
<code>`int`</code>	At least 4	-2147483648 to 2147483647
<code>`long`</code>	At least 4	-2147483648 to 2147483647
<code>`long long`</code>	At least 8	-9223372036854775808 to 9223372036854775807
<code>`float`</code>	4	Approximately $\pm 3.4e38$ (7 decimal digits)
<code>`double`</code>	8	Approximately $\pm 1.7e308$ (15 decimal digits)
<code>`long double`</code>	At least 10	Platform-dependent, greater precision than <code>`double`</code>

Overflow occurs when a value exceeds the maximum representable value for a given data type.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      short int myShortInt = 32767; // Example initialization within the range
6      cout<<myShortInt<<endl;
7      myShortInt = myShortInt+1;
8      cout<<myShortInt<<endl;
9
10     return 0;
11 }
```

Output:

32767
-32768

Underflow occurs when a value becomes smaller than the minimum representable value for a given data type.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5     short int minValue = -32768; // Example initialization within the range
6     cout<<minValue<<endl;
7     minValue = minValue-1;
8     cout<<minValue<<endl;
9
10    return 0;
11 }
```

Output:	-32768
	32767

Identifier (Variable names)

When you try to choose a meaningful variable name.



Identifier (Variable names)



Regardless of which style you adopt, be consistent and make your variable names as sensible as possible. C++ is case sensitive.

- The first character must be one of the letters a through z, A through Z, or an underscore character (_).
- After the first character you may use the letters a through z or A through Z, the digits 0 through 9, or underscores.
- Uppercase and lowercase characters are **different**.

Variable Name	Legal or Illegal?
<code>dayOfWeek</code>	Legal.
<code>3dGraph</code>	Illegal. Variable names cannot begin with a digit.
<code>_employee_num</code>	Legal.
<code>June1997</code>	Legal.
<code>Mixture#3</code>	Illegal. Variable names may only use letters, digits, or underscores.

■ C++ Key words can not be used as Identifier

<code>alignas</code>	<code>const</code>	<code>for</code>	<code>private</code>	<code>throw</code>
<code>alignof</code>	<code>constexpr</code>	<code>friend</code>	<code>protected</code>	<code>true</code>
<code>and</code>	<code>const_cast</code>	<code>goto</code>	<code>public</code>	<code>try</code>
<code>and_eq</code>	<code>continue</code>	<code>if</code>	<code>register</code>	<code>typedef</code>
<code>asm</code>	<code>decltype</code>	<code>inline</code>	<code>reinterpret_cast</code>	<code>typeid</code>
<code>auto</code>	<code>default</code>	<code>int</code>	<code>return</code>	<code>typename</code>
<code>bitand</code>	<code>delete</code>	<code>long</code>	<code>short</code>	<code>union</code>
<code>bitor</code>	<code>do</code>	<code>mutable</code>	<code>signed</code>	<code>unsigned</code>
<code>bool</code>	<code>double</code>	<code>namespace</code>	<code>sizeof</code>	<code>using</code>
<code>break</code>	<code>dynamic_cast</code>	<code>new</code>	<code>static</code>	<code>virtual</code>
<code>case</code>	<code>else</code>	<code>noexcept</code>	<code>static_assert</code>	<code>void</code>
<code>catch</code>	<code>enum</code>	<code>not</code>	<code>static_cast</code>	<code>volatile</code>
<code>char</code>	<code>explicit</code>	<code>not_eq</code>	<code>struct</code>	<code>wchar_t</code>
<code>char16_t</code>	<code>export</code>	<code>nullptr</code>	<code>switch</code>	<code>while</code>
<code>char32_t</code>	<code>extern</code>	<code>operator</code>	<code>template</code>	<code>xor</code>
<code>class</code>	<code>false</code>	<code>or</code>	<code>this</code>	<code>xor_eq</code>
<code>compl</code>	<code>float</code>	<code>or_eq</code>	<code>thread_local</code>	

- A variable must be declared before it is used in the program.
- If the variable is used before declaring it, the compiler will generate an error.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      cout << value; // ERROR! value not defined yet!
6      int value = 100;
7      cout << value; // NO ERROR! inside scope
8
9      return 0;
10 }
```

Constants are values that cannot be modified after they are initialized in a program.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      const float PI = 3.14;
6      PI = 4.0; // ERROR! value can not changed
7
8      return 0;
9  }
```

Brainstorm real-world situations in programming where constants could be applied.

- PI,
- Maximum Number of Users,
- Days of the Week,
- Speed of Light,
- Tax Rates,
- Gravity,
- Maximum File Size,
- Number of Attempts Allowed,
- Discount Percentage

Academic Management System Example

```
#include <iostream>
using namespace std;
int main() {
    // Student record example
    int studentId = 12345;
    string firstName = "Sarah";
    string lastName = "Ahmed";
    char section = 'A';
    double gpa = 3.85;
    bool isActive = true;

    // Course details
    string courseName = "Programming Fundamentals";
    int totalStudents = 30;
    double passingGrade = 60.0;

    return 0;
}
```

(cin) Object



- **cin** is an object in C++ used for reading data from the standard input stream.
- **cin** can be used to read various data types, including integers, floating-point numbers, characters, and strings.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      int age;
6      cout << "Enter your age: ";
7      cin >> age;
8      cout<<"You age = "<<age<<endl;
9
10     return 0;
11 }
```

Output:

```
Enter your age: 19
You age = 19
```

- Multiple input operations can be chained together using the >> operator.

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5      int x,y;
6      cout << "Enter two numbers: ";
7      cin >> x >> y;
8
9      return 0;
10 }
```

Complete example



```
1  #include <iostream>
2  using namespace std;
3  int main() {
4
5     int age;
6     string firstName;
7
8     cout << "Enter your age: ";
9     cin >> age;
10
11    cout << "Enter your First Name: ";
12    cin >> firstName;
13
14    cout<<"Your name is "<<firstName<<" , you are "<<age<<" years old";
15
16    return 0;
17 }
```

Output:

```
Enter your age: 18
Enter your First Name: Karzan
Your name is Karzan, you are 18 years old
```

Thank You

