Tishk International University
Cybersecurity Department
Course Code: CBS 113

Programming Fundamentals

Lecture 2

Variables

Fall 2024

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


Fudgenta
ROSHEN
Fudgenta
Bohem

Communication

Hemin Ibrahim

- hemin.ibrahim@tiu.edu.iq
- (WhatsApp)
  - Sending text messages **ONLY**.
  - Phone calls are **NOT** an option.
  - Voice messages are **NOT** permitted.
- Office: 327

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.

C++ Variables

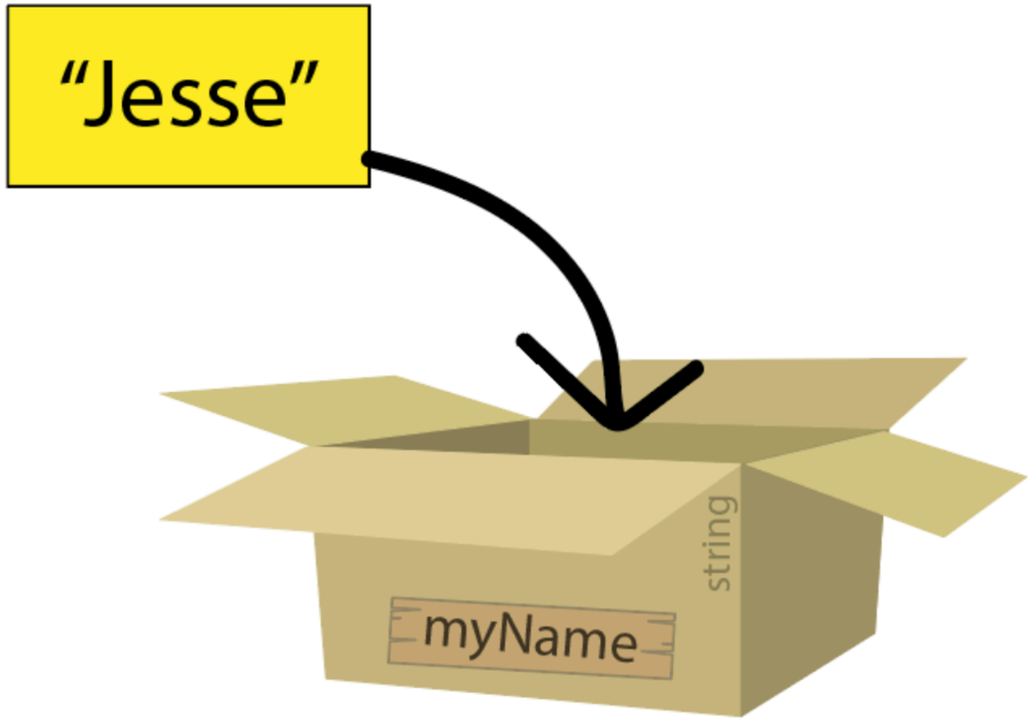
- A named storage location in computer memory
- Like a labeled container that holds a specific type of data
- Has a name, type, value, and memory address

$2x + 1 = 9$

variable

$A = 4 \times s$

variables



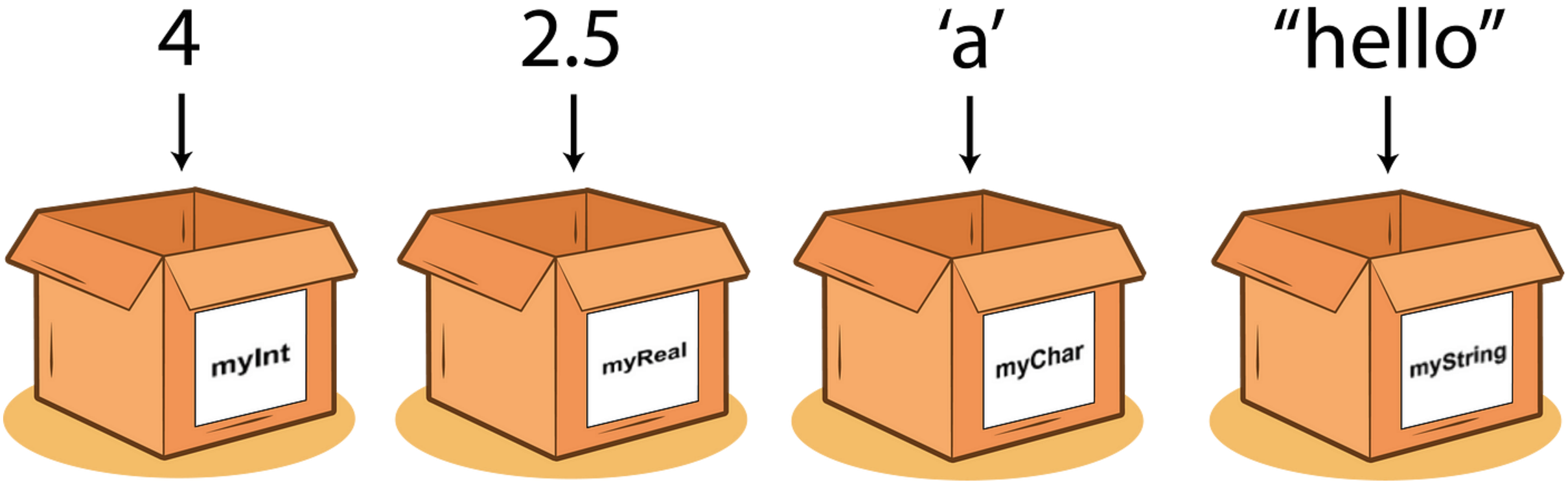
C++ Variables

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

C++ Variables



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

Sample Data Type Matches

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)

C++ Variables

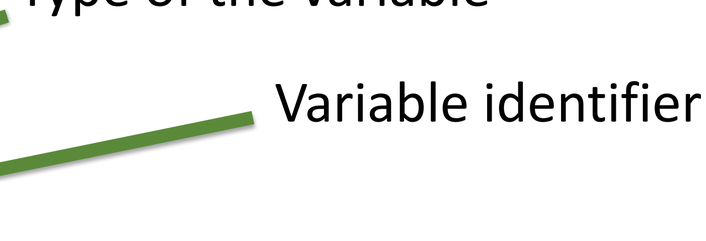
Declaration with initialization

```
data_type variable_name = value;   // Declaration and initialization
data_type variable_name;      // Declaration only
variable_name = value;        // Assignment (initialization)
```

C++ Variables - int

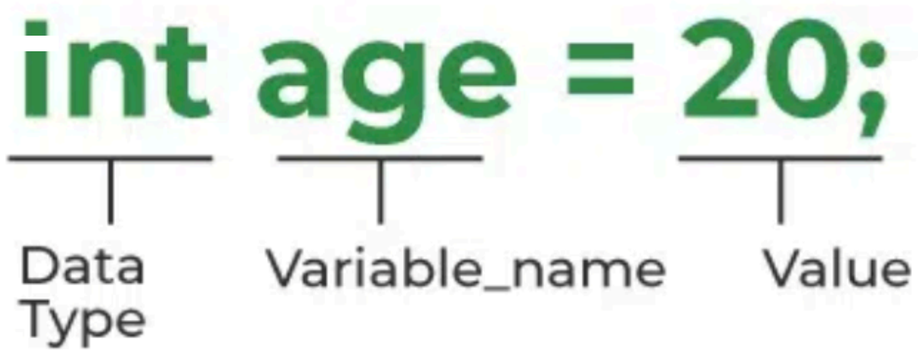
The **int** data type is used for storing whole numbers without decimals. It's suitable for representing quantities that are counted in whole units.

```
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 }
```



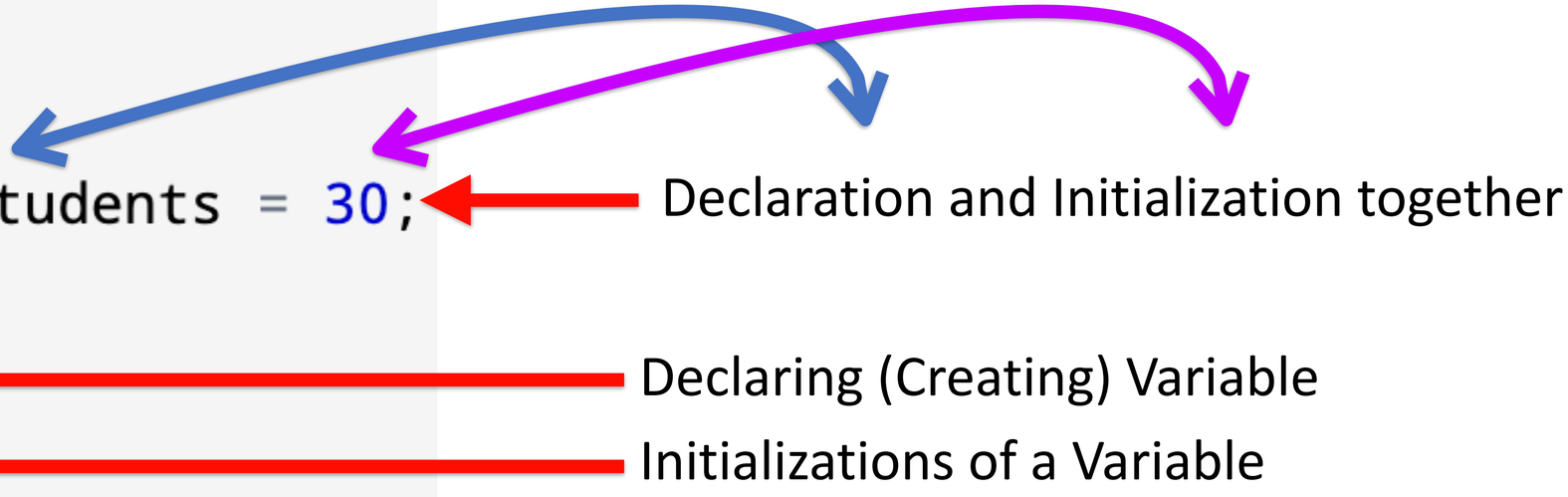
C++ Variables - int

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



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 }
```



Multiple Variables

```
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 todayTemprature = 18.4;
6      float price = 99.9;
7
8      double quiz1;
9      quiz1 = 3.4;
10
11     return 0;
12 }
```

C++ Variables - Floating-point

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 }
```

C++ Variables -string

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 }
```

C++ Variables - Boolean

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;
```

Activity #2

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
`bool`	1	`true` or `false`
`char`	1	-128 to 127 (signed), 0 to 255 (unsigned)
`short`	At least 2	-32768 to 32767
`int`	At least 4	-2147483648 to 2147483647
`long`	At least 4	-2147483648 to 2147483647
`long long`	At least 8	-9223372036854775808 to 9223372036854775807
`float`	4	Approximately ±3.4e38 (7 decimal digits)
`double`	8	Approximately ±1.7e308 (15 decimal digits)
`long double`	At least 10	Platform-dependent, greater precision than `double`

Data types range

Real-World Examples:??

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

Overflow and Underflow

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
```

Overflow and Underflow

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

Identifier

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

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

Variable Scope

- A variable must be declared before it is used in the program.
- If the variable 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

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  }
```

Activity

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

Constant Examples

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

Common Use Cases & Best Practices

Academic Management System Example

```cpp
#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

- 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

```cpp
#include <iostream>
using namespace std;
int main() {

    int age;
    string firstName;

    cout << "Enter your age: ";
    cin >> age;

    cout << "Enter your First Name: ";
    cin >> firstName;

    cout<<"Your name is "<<firstName<<", you are "<<age<<" years old";

    return 0;
}
```

Output:

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

Thank You