CC-301 Programming Fundamentals

Lecture 3

Engr. Waseem Ullah Khan

Programming Style and Documentation

Good programming style and proper documentation make a program easy to read and help programmers prevent errors

- Appropriate Comments and Comment Styles
- Proper Indentation and Spacing

```
cout << 3+4*4;
```
Bad Style

```
cout << 3 + 4 * 4;
```
Good Style

Task: Identify and fix errors in following code:

```
include <iostream>;
using namespace std;

int main
{
    // Display Welcome to C++ to the console
    cout << Welcome to C++! << endl;

    return 0;
}
```

Common Errors

Common Error 1: Missing Braces

```
int main()
{

}
```

← Type this closing brace right away to match the opening brace

Common Error 2: Missing Semicolons

```
int main()
{
      cout << "Programming is fun!" << endl;
      cout << "Fundamentals First" << endl;
      cout << "Problem Driven" << endl
}
```

Missing a semicolon

Common Errors

Common Error 3: Missing Quotation Marks

```
cout << "Problem Driven;
```

Missing a quotation mark

Common Error 4: Misspelling Names

- C++ is case-sensitive. Misspelling names is a common error made by new programmers
- For example, the word main is misspelled as Main in the following code

```
int Main()
{
  cout << (10.5 + 2 * 3) / (45 - 3.5) << endl;
  return 0;
}
```

Class Exercise

Reformat the following program according to the programming style and documentation guidelines

```
#include <iostream>
using namespace std;

int main()
{
cout << "2 + 3 = "<<2+3;
   return 0;
}
```

```
#include <iostream>
using namespace std;

int main ()
{
    cout << "2 + 3 = " << 2 + 3;

    return 0;
}
```

Common Statement

Comments provide information about the program

Comments are for the reader, not the compiler

Two types:

- Single line
  - // This is a C++ program. It prints the sentence:
  - // Welcome to C++ Programming.
- Multiple line
  - /* You can include comments that can
  - occupy several lines. */

```cpp
#include <iostream>
using namespace std;

int main()
{
    // Welcome to the Programming Fundamental Course
    cout << "Fall Semester 2023" << endl;

    /* Welcome to the Programming Fundamental Course
       The Agriculture University Peshawar */
    cout << "Fall Semester 2023" << endl;

    return 0;
}
```

```
Fall Semester 2023
Fall Semester 2023
```

Reserved Words (Keywords)

Reserved words, keywords, or word symbols

- float
- double
- char
- const
- void
- return
- main

What is Identifier

An identifier is name to denote **labels, types, variables, constants or functions**, in a C++ program

C++ is a **case-sensitive** language

- ***Work*** is not the same as ***work***

Identifiers should be descriptive

- Using meaningful identifiers is a good programming practice

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

What is Identifier

Identifiers must be unique

Identifiers cannot be reserved words (keywords) double main return

Identifier must start with a letter or underscore (_), and be followed by letters (A-Z, a-z), digits (0-9), or underscores

Cannot contain blankspaces, commas or special characters ( ) $%&#

Valid

- **age_of_dog** **_taxRateY2K**
- **PrintHeading** **ageOfHorse**

Not Valid

- **age#** **2000TaxRate** **Age-Of-Dog** **main**

What is Identifier

The following are legal identifiers in C++:

- first
- conversion
- payrate

Examples of Illegal Identifiers

Illegal Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

C++ Data Types

In C++, data types are declarations for variables. This determines the ***type*** and ***size*** of data associated with variables. For example,

```
int age = 13;
```

Here, age is a variable of type int. Meaning, the variable can only store integers of either 2 or 4 bytes

- Integer Types
- Character Types
- Bool Types
- Floating Types

C++ Data Types

Integers Types

- **int** keyword is used to indicate integers
- represent whole numbers and their negatives
- Its size is **2 or 4 bytes**
- For example: int balance = 83000;

Character Types

- **char** keyword is used for characters (represent single characters)
- Its size is **1 byte**
- Characters in C++ are enclosed inside single quotes ‘ ’
- For example: char text = ‘h’;

C++ Data Types

Bool Types

- The bool data type has one of two possible values: true or false
- Booleans are used in conditional statements and loops
- For example: bool a = false;

Floating Types

- **float** keyword is used to indicate float
- Used to store floating-point numbers (decimals and exponentials)
- The size of float is 4 bytes
- For example: float area = 64.74;

Samples of C++ Data Values

sample values

0 1 1000 -1 -10 666

sample values

1.0 0.1 95.274 95.0 .265

values

true false

sample values

‘a’ ‘b’ ‘4’ ‘?’ ‘@’

Samples of C++ Data Values

int sample values

0 1 1000 -1 -10 666

float sample values

1.0 0.1 95.274 95.0 .265

bool values

true false

char sample values

‘a’ ‘b’ ‘4’ ‘?’ ‘@’

C++ Type Modifiers

In C++ programming, ***type modifiers*** are used to change the meaning of the fundamental data types

There are four type modifiers in C++:

- short - used for small integers
- Long - used for long integers
- signed
- unsigned

```
// small integer
short a = 12345;

// large integer
long b = 123456;

// positive valued integer
signed int x = 23;

// negative valued integer
signed int y = -13;

// zero-valued integer
signed int z = 0;

// positive valued integer
unsigned int x = 2;
unsigned int y = 0;
```

C++ Data Types

Data Type	Size	Data Storage Range
int	16 bits (2 byte)	-32768 to 32767
short int	16 bits (2 bytes)	-32768 to 32767
long int	32 bits (4 bytes)	-2147483648 to 2147483647
unsigned int	16 bits (2 bytes)	0 to 65535
unsigned long int	32 bits (4 bytes)	0 to 4294967295
float	32 bits (4 bytes)	3.4x10(-38) to 3.4x10(+38)
long float	64 bits (8 bytes)	1.7x10(-308) to 1.7x10(+308)
double	64 bits (8 bytes)	1.7x10(-308) to 1.7x10(+308)
long double	80 bits (10 bytes)	3.4x10(-4932) to 1.1x10(+4932)
char	8 bits (1 byte)	-128 to 127 or 0 to 255

Exercise

Add the correct data type for the following variables:

```
    myNum = 9;
       myDoubleNum = 8.99;
    myLetter = 'A';
    myBool = false;
       myText = "Hello World";
```

```
int myNum = 9;
double myDoubleNum = 8.99;
char myLetter = 'A';
bool myBool = false;
string myText = "Hello World";
```

What is a Variable

A variable is a memory address where data can be ***stored*** and ***changed***

Declaring a variable means specifying both its **name** and its **data type**

Variables, can be declared anywhere in the program

However, they are only visible to program within the block of code in which they are defined

```
int main() {
    int x = 4;
    if (x>3)
    {
        int y = 3;
    }
    return (0)
}
```

Naming Variables

Rules for Variables:

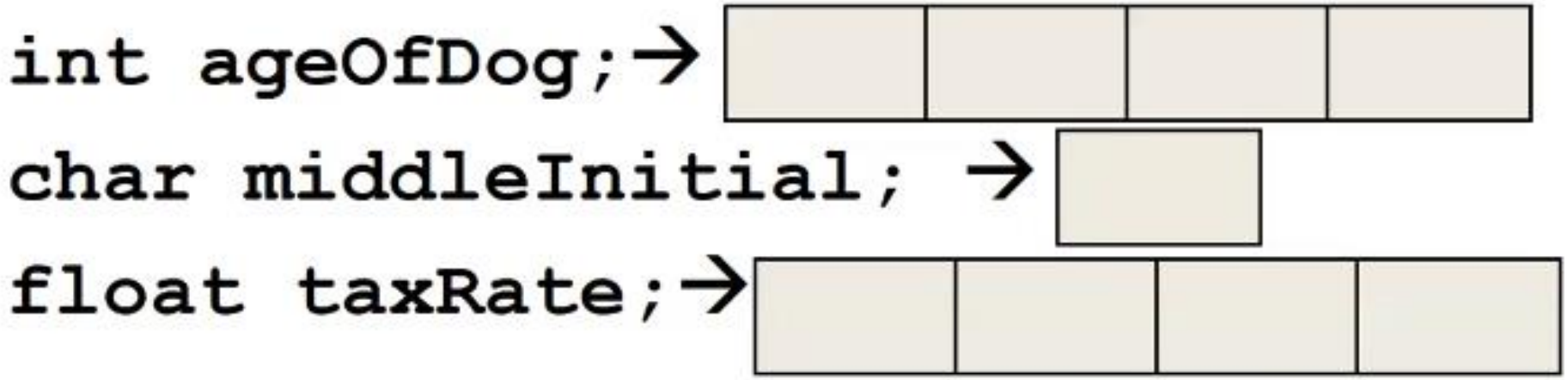
- A variable name can only have alphabets, numbers, and the underscore (_)
- A variable name cannot begin with a number
- Underscore can be used as first character of variable name
- Blank space are not allowed in a variable name
- Begin variable names with a lowercase character. For example, **name** is preferable to **Name**
- A variable name cannot be a *keyword*

```
int a,b;
double d;
/* These are
not good names
*/
```

```
int start_time;
int no_students;
double course_mark;
/* This is a bit better */
```

What Does a Variable Declaration Do?

A declaration tells the compiler to **allocate enough memory** to hold a value of this data type, and to associate the identifier with this location



Variable Declaration

All variables must declared before use

- At the top of the program
- Just before use

Commas are used to separate identifiers of the same type

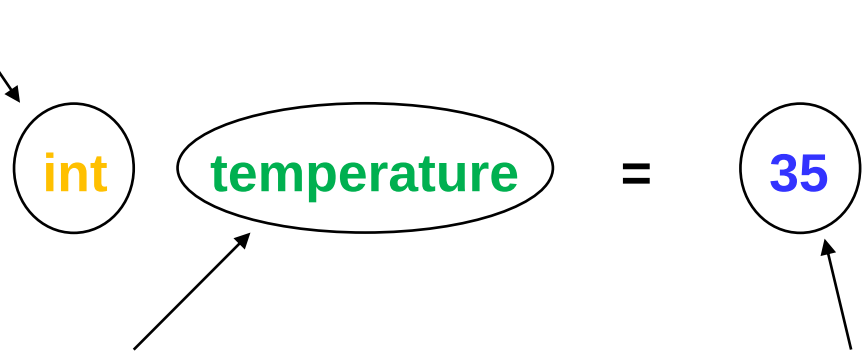
- int count, age;

Variables can be initialized to a starting value when they are declared

- int count = 0;
- int age, count = 0;

An Example of a Variable

Type of the variable is integer (written as “int” in C++)



A name of the variable

An initial value of the variable

Variable Initialization

There are two ways to initialize a variable:

- int feet;

By using the assignment statement

- feet = 35;

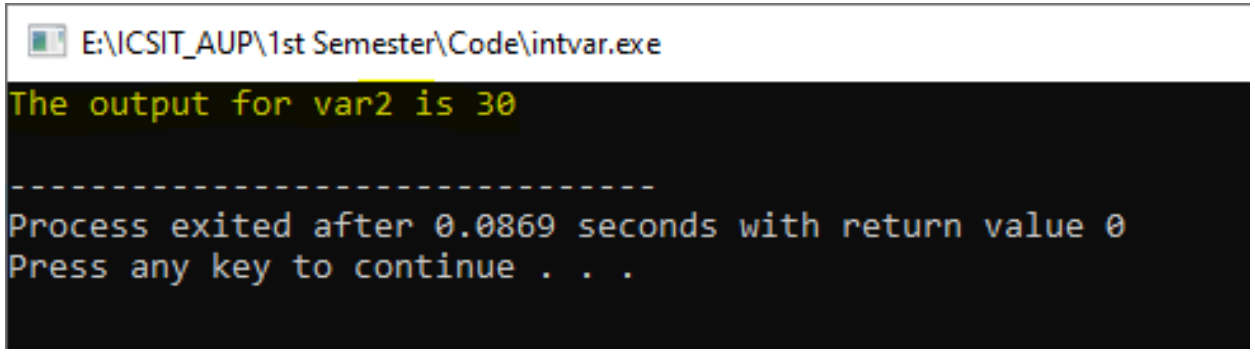
By using a read statement

- cin >> feet;

Example 1

intvar.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int var1;              //define var1
7       int var2;              //define var2
8
9       var1 = 20;             //assign value to var1
10      var2 = var1 + 10;      //assign value to var2
11
12      cout << "The output for var2 is ";       //output text
13      cout << var2 <<endl;                     //output value of var2
14
15      return 0;
16  }
17
```

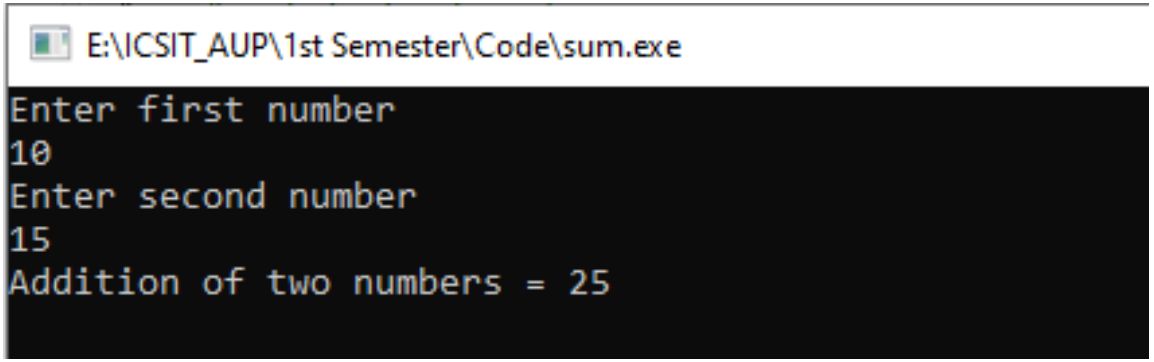


Example 2

Write a C++ code that takes two numbers and displays the addition.

sum.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int a, b, sum;
7       cout <<"Enter first number"<<endl;
8       cin  >>a;
9       cout <<"Enter second number"<<endl;
10      cin  >>b;
11      sum = a + b;
12      cout<<"Addition of two numbers = "<<sum<<endl;
13      return 0;
14  }
```



Example 3

Write a code in C++ that takes radius of a circle as input from user and outputs the circumference and area.

circle.cpp

```cpp
#include <iostream>
using namespace std;

int main(){

    float radius, area, circum;
    float pi = 3.14159;
    cout <<"Enter the radius of circle \n";
    cin  >>radius;

    area = pi * (radius * radius);
    circum = 2 * pi * radius;

    cout <<"The area of circle = "<<area<<endl;
    cout <<"The circumference of circle = "<<circum<<endl;

    return 0;
}
```