Programming Fundamentals (COMP1112)
Lecture 2
Data types and Variables – Input/Output

Division of Science & Technology
University of Education, Lahore.

Data type

- The data type specifies the size and type of information the variable will store

Data type, size, description

long	8 bytes	Stores whole numbers of larger size, without decimals
int	4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits
double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits
boolean	1 byte	Stores true or false values
char	1 byte	Stores a single character/letter/number, or ASCII values

Overflow and Underflow

- Overflow occurs when the value assigned to a variable is more than maximum possible value. (ARIANE 5)
- Underflow occurs when the value assigned to a variable is less than minimum value that could be stored.
- Range of integer: -32768 to 32767

Some examples

- int myNum = 1000;
  cout << myNum;

- float myNum = 5.75;
  cout << myNum;

- double myNum = 19.99;
  cout << myNum;

`float` vs. `double`

- The **precision** of a floating point value indicates how many digits the value can have after the decimal point.
- The precision of `float` is only six or seven decimal digits, while `double` variables have a precision of about 15 digits.
- It is safer to use `double` for most calculations.

Boolean

A boolean data type is declared with the `bool` keyword and can only take the values `true` or `false`.
When the value is returned, `true = 1` and `false = 0`.

- bool ProgramSubmitted = true;
  bool classAttended = false;
  cout <<ProgramSubmitted ; // Outputs 1 (true)
  cout << classAttended ; // Outputs 0 (false)

Character type

- The `char` data type is used to store a **single** character. The character must be surrounded by single quotes, like 'A' or 'c'.
- char myGrade = 'B';
  cout << myGrade;

Alternatively, you can use ASCII values to display certain characters,

ASCII stands for the "American Standard Code for Information Interchange“

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

Output:

ABC

String type

`String` type is used to store a sequence of characters (text). This is not a built-in type, but it behaves like one in its most basic usage. String values must be surrounded by double quotes.

- //include the string library
  #include <string>

  // Create a string variable
  string greeting = "Hello";

  // Output string value
  cout << greeting;

Output:

Hello

Variables

- Variables are containers for storing data values.

Declaring (Creating) Variables

- To create a variable, you must specify the type and name of the variable.

- Syntax

type variable = *value*;

Where type is one of C++ data types and variable is the name of the variable such as “x”, “myVar”

The **equal sign** is used to assign values to the variable.

Some examples

- int myNum = 15; //declare variable and assign value
  cout << myNum;

- int myNum; //declare variable and assign value later
  myNum = 15;
  cout << myNum;

//assigning new value to existing variable with overwrite the previous value

- int myNum = 15; // myNum is 15
  myNum = 10; // Now myNum is 10
  cout << myNum; // Outputs 10

Const keyword

- You can add the const keyword if you don't want others (or yourself) to override existing values (this will declare the variable as "constant", which means **unchangeable and read-only**).

- **const** int myNum = 15; // myNum will always be 15
  myNum = 10; // error: assignment of read-only variable 'myNum'

Other types demonstration

- int myNum = 5; // Integer (whole number without decimals)
- double myFloatNum = 5.99; // Floating point number (with decimals)
- char myLetter = 'D'; // Character
- string myText = "Hello"; // String (text)
- bool myBoolean = true; // Boolean (true or false)

Display Variables

- cout object is used together with the << operator to display variables. To combine both text and a variable, separate them with the << operator (insertion operators <<)

- int myAge = 35;
  cout << "I am " << myAge << " years old.";

Declare Many Variables

- To declare more than one variable of the **same type**, you can use a comma-separated list:

- int x = 5, y = 6, z = 50;
  cout << x + y + z;

Identifiers

- All C++ variables must be identified with the unique names. These unique names are called identifiers.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume)
- It is recommended to use descriptive names in order to create understandable and maintainable code.
- The general rules for constructing names for variables (unique identifiers) are:
- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case sensitive.
- Names cannot contain whitespaces or special characters like !, #, %, etc
- Reserve words (keywords like int) cannot be used as names.

Example program

- A program that takes two integer inputs from user and display its sum.

```
#include<iostream>
using namespace std;
 main()
{
    int x=0;
    int y=0;
    int sum=0;
            cout<< “enter the value for x”; //insertion operators <<
            cin>>x; //extraction operators >>
            cout<< “enter the value for y”;
            cin>>y;
            sum= x+y;
            cout<<“sum of x and y is”<<sum;
}
```

Variable scoping

- Variable is visible within the scope in which it is declared.
- Variable with same name cannot be declared within same scope; however, this can be done if scope is different.
- Formal parameters have functional scope.

```
main()
{
int a=0;
int a=2; //error
}
```

Variable scoping –example 1

```
#include<iostream>
using namespace std;
//global variable
int x=4;
main()
{
        int x=2;  //local variable
        cout<<x;
}
```

Output? How to access global variable?

Variable scoping –example 1

```
#include<iostream>
using namespace std;
//global variable
int x=4;
main()
{
        int x=2;  //local variable
        cout<<x;
        cout<<::x;

}
```

Output:

24

References

- C++ How to Program
  By Deitel & Deitel

- The C++ Programming Language
  By Bjarne Stroustrup

- Object oriented programming using C++ by Tasleem Mustafa, Imran Saeed, Tariq Mehmood, Ahsan Raza

- W3schools.com