



Introduction to Programming

CSC1102 &1103

Lecture-2

American International University Bangladesh
Dept. of Computer Science
Faculty of Science and Information Technology

Lecture 2: Outline

- **Variables, Data Types, and Arithmetic Expressions**
- Working with Variables
 - Understanding Data Types and Constants
 - The Basic Integer Type `int`
 - The Floating Number Type `float`
 - The Extended Precision Type `double`
 - The Single Character Type `char`
 - The Boolean Data Type `_Bool`
 - Storage sizes and ranges
 - Type Specifiers: `long`, `long long`, `short`, `unsigned`, and `signed`
 - Working with Arithmetic Expressions
 - Integer Arithmetic and the Unary Minus Operator
 - The Modulus Operator
 - Integer and Floating-Point Conversions
 - Combining Operations with Assignment: The Assignment Operators
 - Types `_Complex` and `_Imaginary`

Variables

- Programs can use symbolic names for storing computation data
- **Variable: a symbolic name for a memory location**
 - programmer doesn't have to worry about specifying (or even knowing) the value of the location's address
- Variables have to be **declared** before they are used
 - Variable declaration: [symbolic name(*identifier*), type]
- Declarations that reserve storage are called **definitions**
 - The definition reserves memory space for the variable, but doesn't put any value there
- Values get into the memory location of the variable by **initialization** or **assignment**

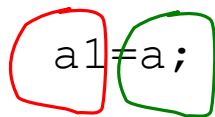
Variables - Examples

```
int a;           // declaring a variable of type int
```

```
int sum, a1,a2; // declaring 3 variables
```

```
int x=7; // declaring and initializing a variable
```

```
a=5; // assigning to variable a the value 5
```

```
a1=a; // assigning to variable a1 the value of a
```

L-value **R-value**

```
a1=a1+1; // assigning to variable a1 the value of a1+1  
// (increasing value of a1 with 1)
```

Variable declarations



The diagram consists of a large black rectangular box. Inside this box, on the left, is a magenta hand-drawn oval containing the text 'Data type'. On the right, also within the black box, is another magenta hand-drawn oval containing the text 'Variable name'. Below the 'Data type' oval is the magenta text 'Which data types are possible'. Below the 'Variable name' oval is the magenta text 'Which variable names are allowed'.

Data type

Which data types
are possible

Variable name

Which variable names
are allowed

Variable names

Rules for valid variable names (*identifiers*) in C/C++ :

- Name must begin with a letter or underscore (`_`) and can be followed by any combination of letters, underscores, or digits.
- Any name that has special significance to the C++ compiler (*reserved words*) cannot be used as a variable name.
- Examples of **valid** variable names: `Sum`, `pieceFlag`, `I`, `J5x7`, `Number_of_moves`, `_sysflag`
- Examples of **invalid** variable names: `sum$value`, `3Spencer`, `int`.
- C++ is case-sensitive: `sum`, `Sum`, and `SUM` each refer to a different variable !
- Variable names can be as long as you want, although only the first 63 (or 31) characters might be significant. (Anyway, it's not practical to use variable names that are too long)
- Choice of meaningful variable names can increase the readability of a program

Data types

- **Basic data types in C++:** `int`, `float`, `double`, `char`, and `bool`.
- Data type `int`: can be used to store integer numbers (values with no decimal places)
- Data type type `float`: can be used for storing floating-point numbers (values containing decimal places).
- Data type `double`: the same as type `float`, only with roughly twice the precision.
- Data type `char`: can be used to store a single character, such as the letter *a*, the digit character *6*, or a semicolon.
- Data type `bool`: can be used to store just the values 0 or 1 (used for indicating a true/false situation). This type has been added by the C99 standard (was not in ANSI C)

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Storage sizes and ranges

- **Every type has a *range* of values associated with it.**
- This range is determined by the amount of storage that is allocated to store a value belonging to that type of data.
- In general, that amount of storage is not defined in the language. It typically depends on the computer you're running, and is, therefore, called *implementation-* or *machine-*dependent.
 - For example, an integer might take up 32 bits on your computer, or it might be stored in 64. You should never write programs that make any assumptions about the size of your data types !
- The language standards only guarantees that a minimum amount of storage will be set aside for each basic data type.
 - For example, it's guaranteed that an integer value will be stored in a minimum of 32 bits of storage, which is the size of a “word” on many computers.

Storage sizes and ranges

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
long int	8bytes	-2,147,483,648 to 2,147,483,647
float	4bytes	Check Book
Double	8bytes	Check Book
long double	12bytes	Check Book

Declaring variables

- Some older languages (FORTRAN, BASIC) allow you to use variables without declaring them.
- Other languages (C, C++, Pascal) impose to declare variables
- **Advantages of languages with variable declarations:**
 - Putting all the variables in one place makes it easier for a reader to understand the program
 - Thinking about which variables to declare encourages the programmer to do some planning before writing a program (What information does the program need? What must the program to produce as output? What is the best way to represent the data?)
 - The obligation to declare all variables helps prevent bugs of misspelled variable names.
 - **Compiler knows the amount of statically allocated memory needed**
 - **Compiler can verify that operations done on a variable are allowed by its type (*strongly typed languages*)**

Declaration vs Definition

- Variable declaration: [Type, Identifier]
- Variable definition: a declaration which does also reserve storage space (memory) !
 - Not all declarations are definitions
 - In the examples seen so far, all declarations are as well definitions
 - Declarations which are not definitions: later in this semester !

Assigning values to char

```
char letter; /* declare variable letter of type char */

letter = 'A'; /* OK */
letter = A;   /* NO! Compiler thinks A is a variable */
letter = "A"; /* NO! Compiler thinks "A" is a string */
letter = 65;  /* ok because characters are really
               stored as numeric values (ASCII code),
               but poor style */
```

Variables and Data Types Examples

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

int main() {
    string name = "RIFATH MAHMUD";
    string id = "15-XXXX-1";
    int age = 50;
    float cgpa=3.96;
    float height=5.7;

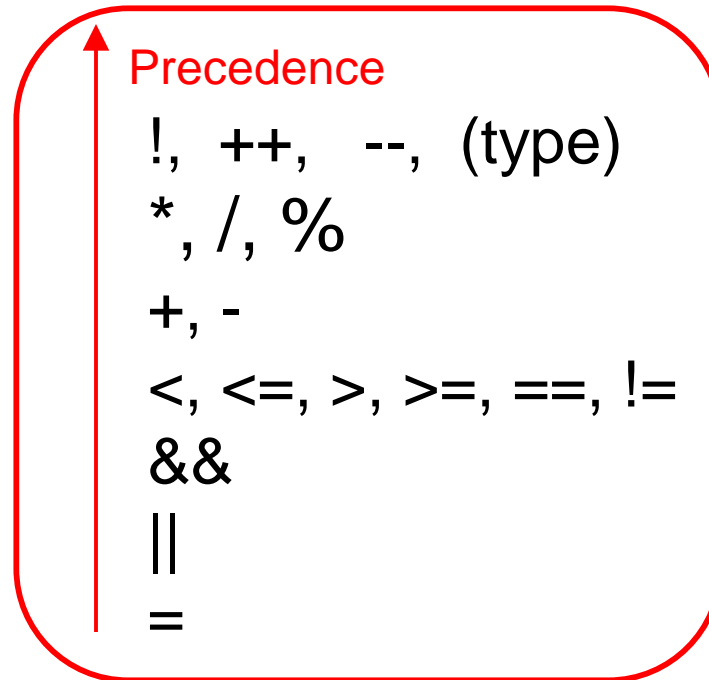
    cout<<"NAME\t:"<<name<<"\n";
    cout<<"ID\t:"<<id<<"\n";
    cout<<"AGE\t:"<<age<<" years\n";
    cout<<"CGPA\t:"<<cgpa<<"\n";
    cout<<"HEIGHT\t:"<<height<<"\n";

    return 0;
}
```

Working with arithmetic expressions

- Basic arithmetic operators: +, -, *, /
- **Precedence**: one operator can have a higher priority, or *precedence*, over another operator.
 - Example: * has a higher precedence than +
 - $a + b * c$
 - if necessary, you can always use parentheses in an expression to force the terms to be evaluated in any desired order.
- **Associativity**: Expressions containing operators of the same precedence are evaluated either from left to right or from right to left, depending on the operator. This is known as the *associative* property of an operator
 - Example: + has a *left to right* associativity
- **Check Book (Page 29) for Details**

Precedence of operators



Working with arithmetic expressions

```
#include <iostream>
using namespace std;
int main (void)
{
    int a = 100;
    int b = 2;
    int c = 25;
    int d = 4;
    int result;
    result = a - b; // subtraction
    cout<<"a - b = "<< result<<endl;
    result = b * c; // multiplication
    cout<<"b * c = " << result<<endl;;
    result = a / c; // division
    cout<<"a / c = " << result<<endl;
    result = a + b * c; // precedence
    cout<<"a + b * c = " << result<<endl;
    return 0;
}
```

Integer and Floating-Point Conversions

- Assign an integer value to a floating variable: does not cause any change in the value of the number; the value is simply converted by the system and stored in the floating
- Assign a floating-point value to an integer variable: the decimal portion of the number gets truncated.
- Integer arithmetic (division):
 - int divided to int => result is integer division
 - int divided to float or float divided to int => result is real division (floating-point)

Solving Quadratic Equation using C++

```
#include<iostream>
#include<math.h>
using namespace std;

int main() {

    float a,b,c;

    cout<<"Input The Value of a \t= ";
    cin>>a;
    cout<<"Input The Value of b \t= ";
    cin>>b;
    cout<<"Input The Value of c \t= ";
    cin>>c;
    float x1,x2;
    x1= (-b+sqrt(pow(b,2)-4*a*c))/(2*a);
    x2= (-b-sqrt(pow(b,2)-4*a*c))/(2*a);
    cout<<"Solution x1 = "<< x1 <<endl;
    cout<<"Solution x2 = "<< x2 <<endl;

    return 0;
}
```

SOME SAMPLE QUESTIONS

- What is compiler and why do we need it ?
- What are the rules of valid Variable Name.
- What is ASCII table? Why do we need it?
- Why do we need data types and explain the basic data types.
- How to take input and print the out put in C++
- What is operator, operand and operation?
- What are the basic arithmetic operation?
- Explain precedence of operator in C++
- What are the basic relational operation?
- What is the purpose of using % operator, which data type is valid for using % operator.

SOME SAMPLE QUESTIONS

- Explain the integer division and floating point division.
- What are the functions and library required for writing Square root of a number and Power of a number. Show how to properly use them.
- **ANY TOPIC COVERD IN THEORY AND LAB CLASS ARE EQALLY IMPORTAN, CHECK THE NOTICE IN NEXT PAGE**

!!! IMPORTANT !!!

**Remember these are not the only
questions**

**it is for helping you to think how you
should be prepare yourself for the exams
read Books class notes to take
preparation**