Al-Mustaqbal University

College of Engineering Technology

Cybersecurity Techniques Engineering Department

Programming Essential

Lecture 3

Variables, Data Types, Declaration of variables, Constants, Statements, and Operators

PhD. BEng. Ahmed Hasan Janabi
PhD in Cybersecurity
Email: Ahmed.Janabi@uomus.edu.iq

Lab Objectives

By the end of this lab, students will be able to:

Identify variables, data types, and constants

Use correct naming rules for variables

Perform basic input and output operations

Write simple C++ programs that store and display data

Keywords and Identifier

- Keywords are predefined, reserved words used in programming that have special meanings to the compiler.
- Keywords are part of the syntax and they cannot be used as an identifier.
- For example:

```
int money;
```

- Here, int is a keyword that indicates money is a variable of type int (integer).
- Since C++ is a **case sensitive language**, all keywords must be written in lowercase.

A ≠ a
X ≠ x
Case ≠ case

Note:
case-sensitive for both variable names and function names

Keywords and Identifier

❖ The table below lists some of the keywords allowed in ANSI C.

C++ Keywords			
auto	default	goto	struct
break	else	long	switch
case	enum	return	typedef
char	extern	register	union
const	for	signed	unsigned
continue	float	sizeof	void
do	if	short	volatile
double	int	static	while

Note:

As can be seen in the table above, all the commands (keywords) should be written in lowercase (small letters).

Keywords and Identifier

- Identifier refers to name given to entities such as variables, functions, structures etc.
- Identifiers must be unique. They are created to give a unique name to an entity (variable,const, etc.) to identify it during the execution of the program.
- For example:

```
int money;
double accountBalance;
```

  - Here, money and accountBalance are identifiers.
- Always remember that identifier names must be different from keywords.
  - You ***cannot*** use `int` as an identifier because `int` is a keyword.

Case Sensitivity

- C ++ language is a case-sensitive.
- It differs whether an identifier, such as a variable name, is uppercase or lowercase

Examples:

- area
- Area
- AREA
- ArEa

all are seen different variables by the compiler.

Rules for naming Identifiers

- A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
- Sperate “words” within a variable name using underscores and mixed upper and lower case.

  Examples:
  - ✓ surfaceArea
  - ✓ surface_Area
  - ✓ surface_area

- The first letter of a variable name must be either a letter or an underscore, i.e., variable names cannot begin with a number.
- You cannot use any reserved word (keywords) like int, char etc. as a variable name.

Data types in C++ Programming

- In C++ programming, data types are declarations for variables. This determines the type and size of data associated with variables.
- The following table contains some commonly used data types in C++ programming.

Table: Basic types of data in C programming

Type	Size (bytes)	Format Specifier
int	at least 2, usually 4	%d
char	1	%c
float	4	%f
double	8	%lf
short int	2 usually	%hd
unsigned int	at least 2, usually 4	%u
long int	at least 4, usually 8	%ld

Declaring Variables

- Before you use a variable, you should first declare it, so the compiler can know it.
- Declaring a variable means specifying the data type of the variable

***Example** of variable declarations*

```
int  myVar;
```

- myVar is a variable of `int` (integer) type.
- the size of `int` is 4 bytes.

Variables

- In programming, a variable is a container (storage area) to hold data.
- To indicate the storage area, each variable should be given a unique name (identifier).
- Variable names are just the symbolic representation of a memory location.

- For example:

```
int playerScore = 95;
```

  - In the above definition, playScore is a variable of `int` type.
  - The variable is assigned to an integer value 95
  - As the name indicates, the value of a variable can be changed through the program.

```
char ch = ‘a’;
// in somewhere in the code, you write
ch = ’1’;
```

Rules for naming Variables

Note:

- You should always try to give meaningful names to variables.
- For example:

  `firstName` is a better variable name than `fn`.

- C++ is a strongly typed language. This means that the *variable* ***type*** **cannot** be changed once it is declared.
- For example:

```
int  number = 5; // integer variable
number = 5.5; // error
double  number ; // error
```

- In the above, we define the type of number as `int`.
- Thus, we *cannot* assign a *floating-point* (decimal (عدد عشري value 5.5 to an `int` variable.
- Additionally, we cannot redefine the same variable with different data type (i.e., double).
- A *floating-point* variable should be declared (defined) either as `double` or `float` to store its decimal value in C++

Constants

- ❖ If you want to define a variable whose value cannot be changed, you can use the const keyword.
- ❖ This will create a constant, for example:

```
const double PI = 3.14;
```

Note:

- ➢ the keyword `const` have added
- ➢ Also, PI is a symbolic constant; its value *cannot be changed*.

```
const double PI = 3.14 ; // define PI as a constant with value 3.14
```

- ➢ Then somewhere in your code you write the blow

```
PI = 2.9; // error
```

Example of Declarations and Assignments

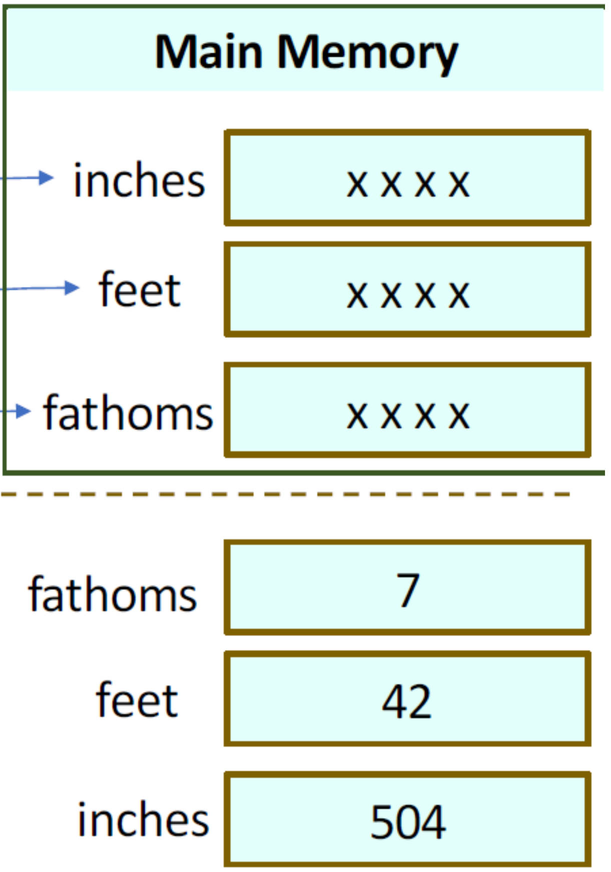
- When you declare a variable with a data type, a space inside the memory is set (reserved) by a unique address to hold the value of the defined variable.

- The address of the space is associated with the variable name

Example:

```c
#include <stdio.h>

int main() {
int inches, feet, fathoms;
fathoms = 7;
feet = 6 * fathoms;
inches = 12 * feet;
```



C output – printf

- In C++ programming, printf() is one of the main output function.
- The function sends formatted output to the screen.
- For example

```
#include <stdio.h>
int main()
{
    //display the string inside quotations
    printf("C++ Programming");
    return 0;
}
```

Output

```
C++ Programming
```

Integer Output – printf

❖The following C++ program outputs an integer number on the screen.

```
#include <stdio.h>
int main()
{
    int testInteger = 5;
    printf("Number = %d", testInteger);
    return 0;
}
```

Output

```
Number = 5
```

Note:

- The %d format specifier is used to print `int` data types.
- Here, the %d inside the quotations will be replaced by the value of `testInteger`.

Print (Output) float and double

❖ The following C++ program outputs a float and a double number on the screen.

```
#include <stdio.h>
int main()
{
    float num1 = 13.5;
    double num2 = 12.4;
    printf("num1 = %f\n", num1);
    printf("num2 = %lf", num2);
    return 0;
}
```

Output

```
num1 = 13.500000
num2 = 12.400000
```

Note:

- ➢ To print a float number, `%f` format specifier is used.
- ➢ Similarly, we use `%lf` to print double values.

Print Characters – printf

❖ The following C++ program prints a character on the screen.

```
#include <stdio.h>
int main()
{
  char chr ='a';
  printf("Character = %c", chr);
  return 0;
}
```

Output

```
Character = a
```

Note:

➢ To print char, `%c` format specifier is used.

C input – scanf

- ❖In C++ programming, scanf() is one of the commonly used input function
- ❖The scanf() function is used to read input variables entered by users from the keyboard.

```
#include <stdio.h>
int main()
{
    int testInteger;
    printf("Enter an integer: ");
    scanf("%d", &testInteger);
    printf("Number = %d", testInteger);
    return 0;
}
```

Output

```
Enter an integer: 4
Number = 4
```

C input – scanf

```
scanf("%d", &testInteger);
printf("Number = %d", testInteger);
```

- Here, the `%d` format specifier is used inside the `scanf()` function to take int input from the user.
- When the user enters an integer, it is stored in the **testInteger** variable.
- Also, notice, that we have used **&testInteger** inside `scanf()`

  → because **&testInteger** gets the *address* of **testInteger**, and the value entered by the user is stored in that address.

Format Specifiers for I/O

❖ As can noticed from the previous examples, we have used the following format specifiers:

- **%d** for **int**
- **%f** for **float**
- **%lf** for **double**
- **%c** for **char**

Scan (Input) float and double

- The following C++ program inputs a float and double number from the keyboard and the display them on the screen.

```
#include <stdio.h>
int main()
{
    float num1;
    double num2;
    printf("Enter a number: ");
    scanf("%f", &num1);
    printf("Enter another number: ");
    scanf("%lf", &num2);
    printf("num1 = %f\n", num1);
    printf("num2 = %lf", num2);
    return 0;
}
```

Output

```
Enter a number = 12.523
Enter another number = 10.2
num1 = 12.523000
num2 = 10.200000
```

Input/Output a character

❖ The below c++ program inputs a char from the keyboard, then displays (outputs) its ASCII on the screen

```
#include <stdio.h>
int main()
{
    char chr;
    printf("Enter a character: ");
    scanf("%c", &chr);
    // When %c is used, a character is displayed
    printf("You entered %c.", chr);
    // When %d is used, ASCII value is displayed
    printf("ASCII value is %d", chr);
    return 0;
}
```

Output

```
Enter a character: w
You entered w.
ASCII value is 120
```

I/O multiple values

❖ Take multiple values form the user and display them on the screen

```c
#include <stdio.h>
int main()
{
    int a;
    float b;
    printf("Enter an integer and then a float: ");
    scanf("%d%f", &a, &b);
    printf("You entered %d and %f", a, b);
    return 0;
}
```

Output

```
Enter an integer and then a float: -3
3.4
You entered -3 and 3.400000
```

THANK
YOU