

# INTRODUCTION TO COMPUTER SCIENCE

---

Dr. Yasmine El-Glaly

Fall 2013

# Ch.6: Programming Languages

- Traditional Programming Concepts
- Procedural Units

# Programming

"Learning to write programs stretches your mind, and helps you think better, creates a way of thinking about things that I think is helpful in all domains."

-Bill Gates

Public School 1st graders i



The birthplace  
of Skype

One o  
g



# Programming



We need more programmers!

OVER THE NEXT 10 YEARS



1/10

Yet, only 1 in 10 schools currently teach computer science

## How Programming Jobs Compare

Of The Top 10 Jobs of 2013,  
4 require programming experience



Leaving  
**1 million**  
empty jobs!



**60%** of math and science jobs are computing jobs

**BUT ONLY**

**2%** of math and science students are computer science students



# Traditional Programming

Procedural Units

- A program consists of a collection of statements
  - **Declarative statements** define customized terminology that is used later in the program, such as the names used to reference data items;
  - **imperative statements** describe steps in the underlying algorithms;
  - **comments**

# Variables and Data Types

- High-level programming languages allow locations in main memory to be referenced by names called: **variable**
- the type of data that will be stored at the memory: **data type** e.g. integer, float, character, Boolean
- Examples:

```
int Height, Width;
```

```
int WeightLimit = 100;
```

```
char Letter, Digit;
```

# Array

- A block of elements of the same type such as a one-dimensional list, a two-dimensional table with rows and columns, or tables with higher dimensions.
- **Indices**
- Example:

```
int Scores[2][9];
```

**Scores**


Scores [1] [3] in C  
and its derivatives  
where indices start  
at zero.

# Structure

- A block of data in which different elements can have different types.

```
struct {char   Name[25];  
        int    Age;  
        float  SkillRating;}  
Employee;
```

- a programmer can use the structure name (Employee) to refer to the entire aggregate
- or can reference individual **fields** within the aggregate by means of the structure name followed by a period and the field name (such as Employee.Age).



# Traditional Programming

Procedural Units

- Constants
  - A fixed, predetermined value
  - Example:

```
const int AirportAlt = 645;
```

# Traditional Programming

Procedural Units

- Assignment Statements
  - The most basic imperative statement is the **assignment statement**
  - **Example:**
$$Z = X + Y;$$
  - Rules of **operator precedence**
$$2 * 4 + 6 / 2$$
  - parentheses
$$2 * (4 + 6) / 2$$

## Traditional Programming

Procedural Units

- Overloading
  - multiple use of an operation symbol
  - Ex.  
"abra" + "cadabra"
- is *abracadabra*.

# Traditional Programming

Procedural Units

- Control Statements

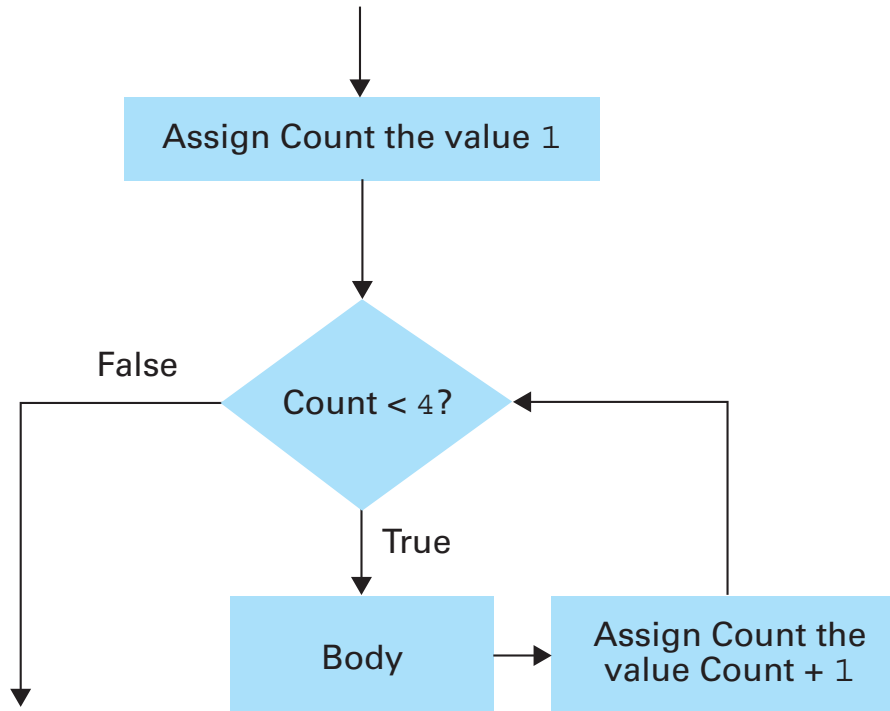
```
if (condition) statementA  
    else statementB;
```

```
switch (variable) {  
    case 'A': statementA; break;  
    case 'B': statementB; break;  
    case 'C': statementC; break;  
    default: statementD}
```

# Traditional Programming

Procedural Units

- Control Statements



```
for (int Count = 1; Count < 4; Count++)  
    body ;
```

```
while (condition)  
    {loop body}
```

# Traditional Programming

Procedural Units

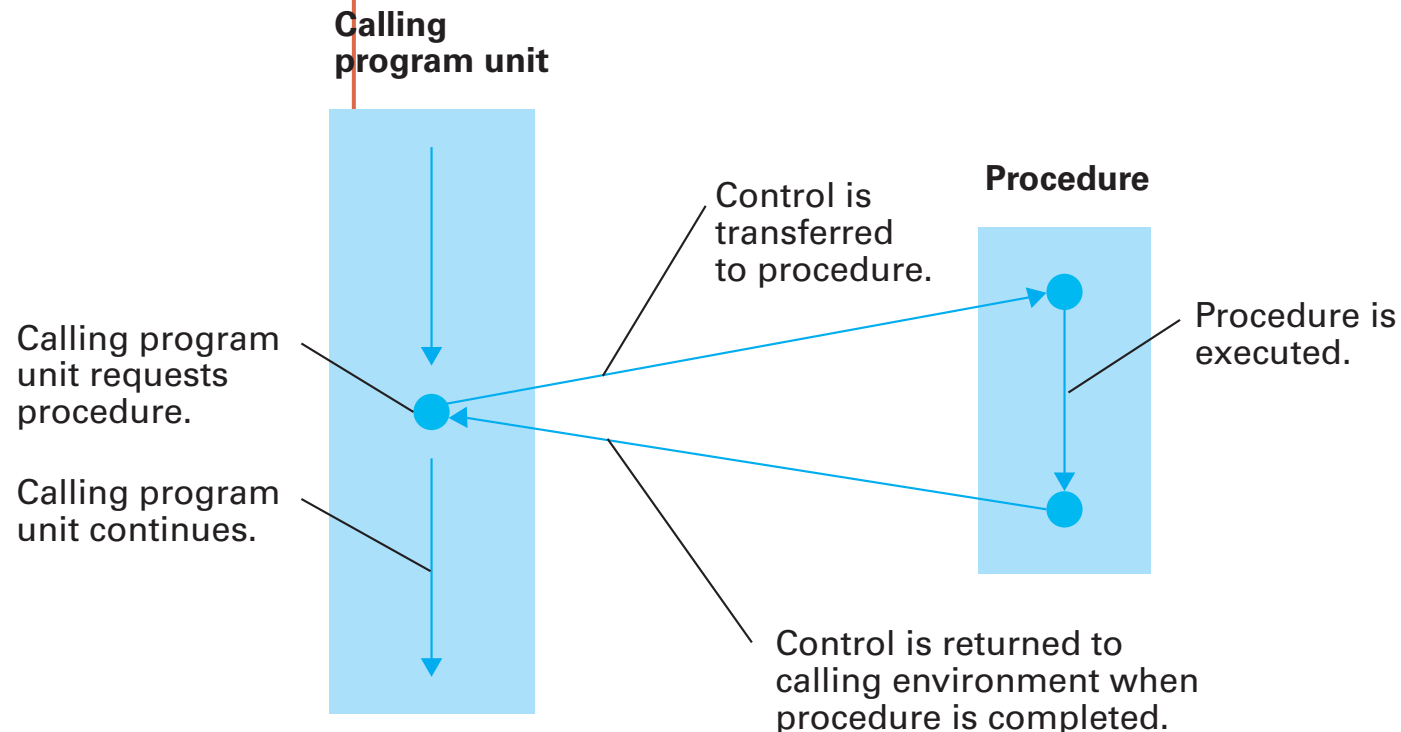
- Comments
  - explanatory statements
  - for internal documentation,
  -

```
/* This is a comment. */
```

```
// This is a comment.
```

## Procedural Units

- Procedure
  - a set of instructions for performing a task that can be used as an abstract tool by other program units.



## Procedural Units

- Procedure
  - **procedure's header:**
    - identifies, among other things, the name of the procedure.
  - Following this header are the statements that define the procedure's details.
  - a variable declared within a procedure is a **local variable**,
  - The portion of a program in which a variable can be referenced is called the **scope** of the variable
  - **global variables**



# • Parameters

## Procedural Units

Starting the head with the term “void” is the way that a C programmer specifies that the program unit is a procedure rather than a function. We will learn about functions shortly.

The formal parameter list. Note that C, as with many programming languages, requires that the data type of each parameter be specified.

```
void ProjectPopulation (float GrowthRate)
```

```
{ int Year;
```

This declares a local variable named Year.

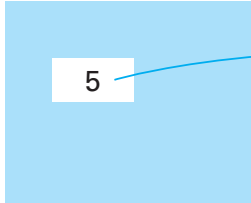
```
Population[0] = 100.0;  
for (Year = 0; Year <= 10; Year++)  
Population[Year+1] = Population[Year] + (Population[Year] * GrowthRate);  
}
```

These statements describe how the populations are to be computed and stored in the global array named Population.

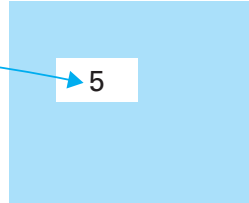
Executing the procedure Demo and passing parameters by value

- a. When the procedure is called, a copy of the data is given to the procedure

Calling environment



Procedure's environment



- b. and the procedure manipulates its copy.

Calling environment

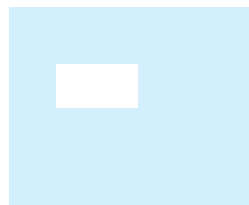


Procedure's environment



- c. Thus, when the procedure has terminated, the calling environment has not been changed.

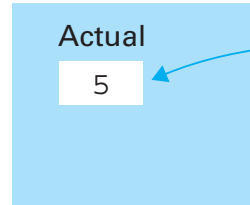
Calling environment



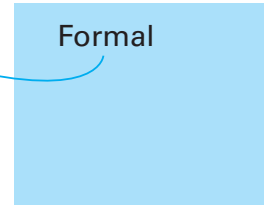
Executing the procedure Demo and passing parameters by reference

- a. When the procedure is called, the formal parameter becomes a reference to the actual parameter.

Calling environment

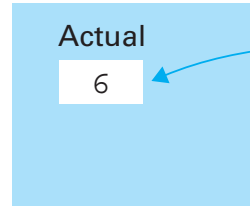


Procedure's environment

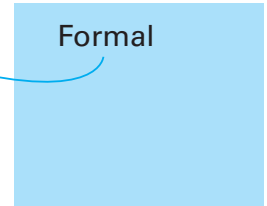


- b. Thus, changes directed by the procedure are made to the actual parameter

Calling environment

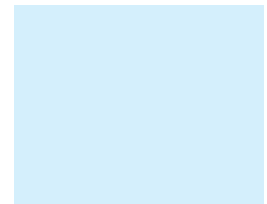


Procedure's environment



- c. and are, therefore, preserved after the procedure has terminated.

Calling environment



# Function

- The term **function** refers to a program unit similar to a procedure except that a value is transferred back to the calling program unit as “the value of the function.”

The function header begins with the type of the data that will be returned.

```
float CylinderVolume (float Radius, float Height)
```

```
{ float Volume;
```

Declare a local variable named Volume.

```
Volume = 3.14 * Radius * Radius * Height;
```

```
return Volume;
```

Compute the volume of the cylinder.

```
}
```

Terminate the function and return the value of the variable Volume.

# Assignment

- 10 programming problems to be solved at the Lab.

# Notes about the exam

- Lab
  - Computer-based
- Oral
  - Discussion question
- Final
  - MCQ + problems + programming, (NO articles questions)
- Book
  - Ch.1 except 1.7
  - Ch.2
  - Ch. 3 except 3.1
  - Ch. 4
  - Ch. 6 (6.2 and 6.3)

# Notes to You

- How to start reading a book?
  - Read the contents
  - Skim each chapter
  - Prepare a side list with your translated words
  - Learn the meanings of the scientific expressions

Have you noticed that Appendix F is Answers to the chapters questions?

I think we underrate students if we assume that we have to explain everything in class. We should be helping them learn to learn on their own.

# Students' feedback

- Comments about the book
  - Writing style, arrangement, examples, problems
- Comments about the lecture
  - Timing, coverage, explanation, respond to questions
- Comments about the slides
  - Clarity, animations, videos,
- Comments about the assignments
  - Variety, coverage, amount, time allowance
- Comments about the Lab
  - Computers, software, discipline,

Thank y all n c  
u nxt year