# DASF004
# Basic and Practice in Programming

## Lecture 1
## Introduction

# Agenda

Administration
Food for your MIND

How a computer works
A brief history of computers
Compiler
Programming in C

# Administration

## Issues with lectures video

## Lectures also available on Youtube!

https://www.youtube.com/channel/UC90vpI4GEDfmZM5oc8Vo8Kw

Subtitle is available (automatically generated by youtube)!!!

## Structure of Lectures and Labs

# Structure of Lectures

- Each Lecture is 75 minutes
- One Lecture per week
- 
- Each Lecture will start off with a “Food for your MIND” Section (about 5 minutes)
- Not related to course material
- About cutting edge research in Science and Technology

- Lecture materials will elaborate the knowledge and theory
-

# Structure of Labs

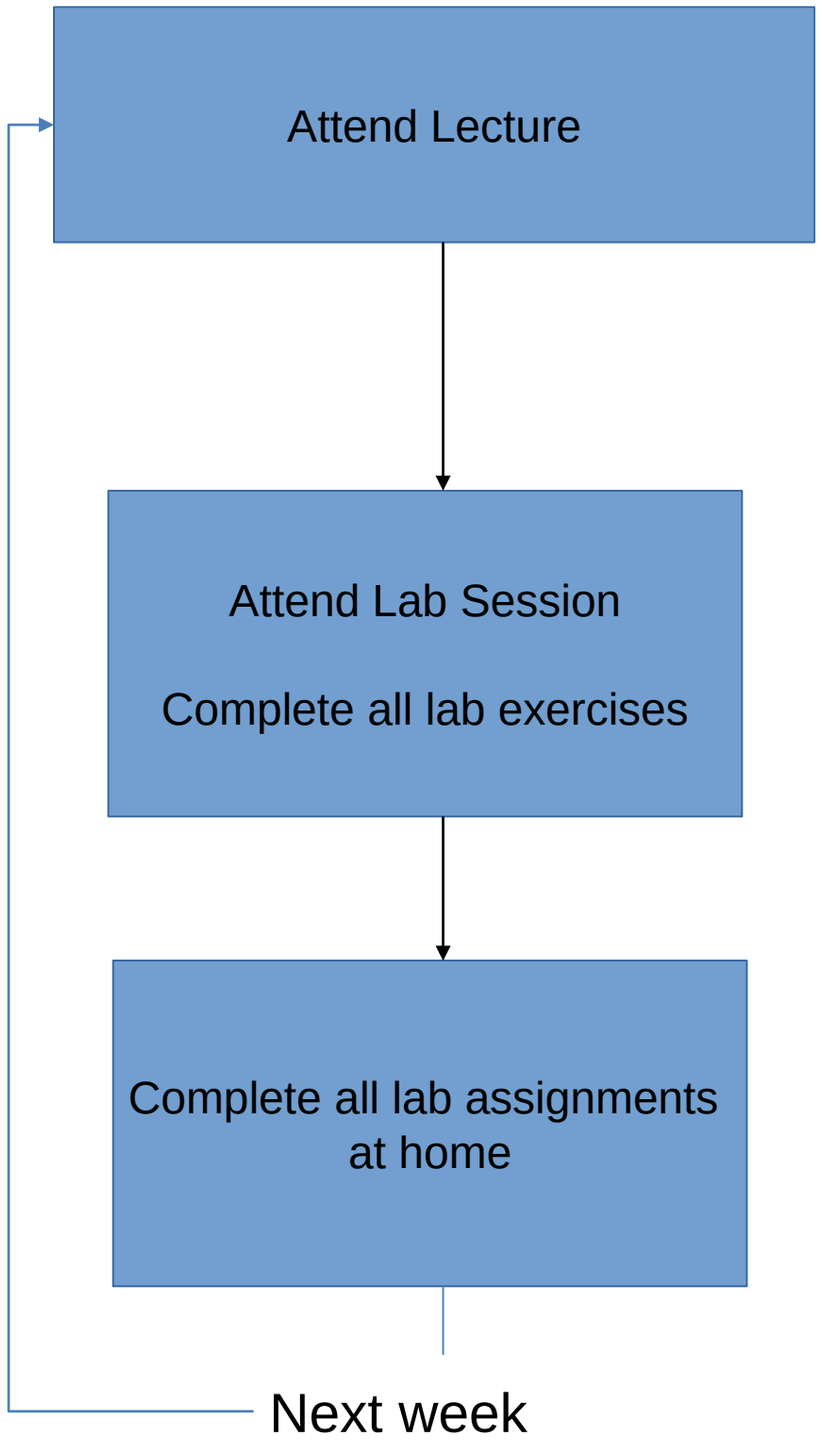
- Each lab session is 75 minutes
- One lab session per week
- Video of the lab will be uploaded to iCampus Wednesday 12:00 pm
- TAs will be available on the discussion board during the lab session
  - You may ask questions to the TA in the discussion board
- Lab materials will focus on practical programming
- Each lab session consists of lab exercises and homework assignment

# Structure of Labs

- Lab Exercises – to be completed during lab day
  - Submit your lab exercises through iCampus
  - You have to submit it before the end of day (Wednesday)
    - (if you don’t submit, you will not have the score)

- Homework Assignment – You can complete it at home
  - Submit your Homework assignments through iCampus
  - You have 1 week to complete it
  - Due date: Tuesday 23:59 pm next week

# Road-map for each week ...




| | |
|---|---|
| Monday | Attend lecture |
| Tuesday | |
| Wednesday | Attend lab, complete lab exercise<br>Homework assignment is assigned, Due date is Tuesday |
| Thursday | |
| Friday | |
| Saturday | |
| Sunday | |
| Monday | Attend lecture |
| Tuesday | Due date for Homework Assignment |
| Wednesday | Attend lab, complete lab exercise<br>Homework assignment is assigned, Due date is Tuesday |
| Thursday | Grade for last Lab exercise is on iCampus |
| Friday | |
| Saturday | |
| Sunday | |
| Monday | Attend lecture |
| Tuesday | |
| Wednesday | Grade for last Homework Assignment is on iCampus<br>Attend lab, complete lab exercise<br>Homework assignment is assigned, Due date is Tuesday |

Q&A

A: I heard there is a mac version for Dev C++. You may also run windows software using Wine emulator. I don't have a mac machine to test them out, but my previous students said they work okay.

Q: Do I have to take part in the Discussion Board?

A: No, if you don’t have any question.

# Q&A

Q: Can I use IDE other than Dev C++ like Visual Studio?
A: Yes, you may, but we'll use Dev C++ in the instruction.

# Q&A

Q: During the lab session, do I have to take class on time?
A: You may take the lab session anytime. But notice that the due time for the lab exercise is 23:59 pm. If you watch the lab video late, you’ll have less time to complete the lab exercise.

# You should submit your source code only!

- Source code
  - Your code written in C!
  - The “XXXXXX.c” or “XXXXXX.cc” file
- No grade will be given if you submit the output of your program
- No grade will be given if you submit the screen capture of your source code
- Double check your submission
  - Make sure you submit the correct file!

# Grades Reporting

- Grades will be available
  - About one week after the deadline (for lab exercises and homework assignments)
  - About one week after you submit it (for attendance)
- You should wait about one week for your grade
  - It takes about one week for us to grade
  -

# Food for your MIND!

- Augmented Reality Application
- Research Question:
  - How to direct user’s attention in an omni-directional workspace?
- Directional Cue:
  - Up, down, left, right, front, back, …
- Visual Highlight
- Problems?
- How to solve these problems?

# Computers?

- A general-purpose device that can be **programmed** to carry out a finite set of **arithmetic** or **logical** operations.
- Since a sequence of operations can be readily changed, the computer can solve more than one kind of problem.

# Computer

Computer = Hardware + Software

Like piano and music

- “Hardware” refers to the physical parts of the computer (cf. “software” refers to the code that runs on the computer)

## Computers

- What a computer can do?
  - Everything when you can explain exactly how to do
  - E.g., “check if a number is a prime”
- What a computer cannot do?
  - Everything that you cannot explain exactly how to do
  - E.g., Feeling, thinking, …

# Problem Solving with Computer

- Check if a number is a prime.
- Tell me how to determine step by step.

```
2, 3, 5, 13, 89, 233, 1597, 28657,
514229, 433494437, 2971215073,
99194894755497,
1066340417491710595815721 69,
…
```

# Problem Solving with Computer

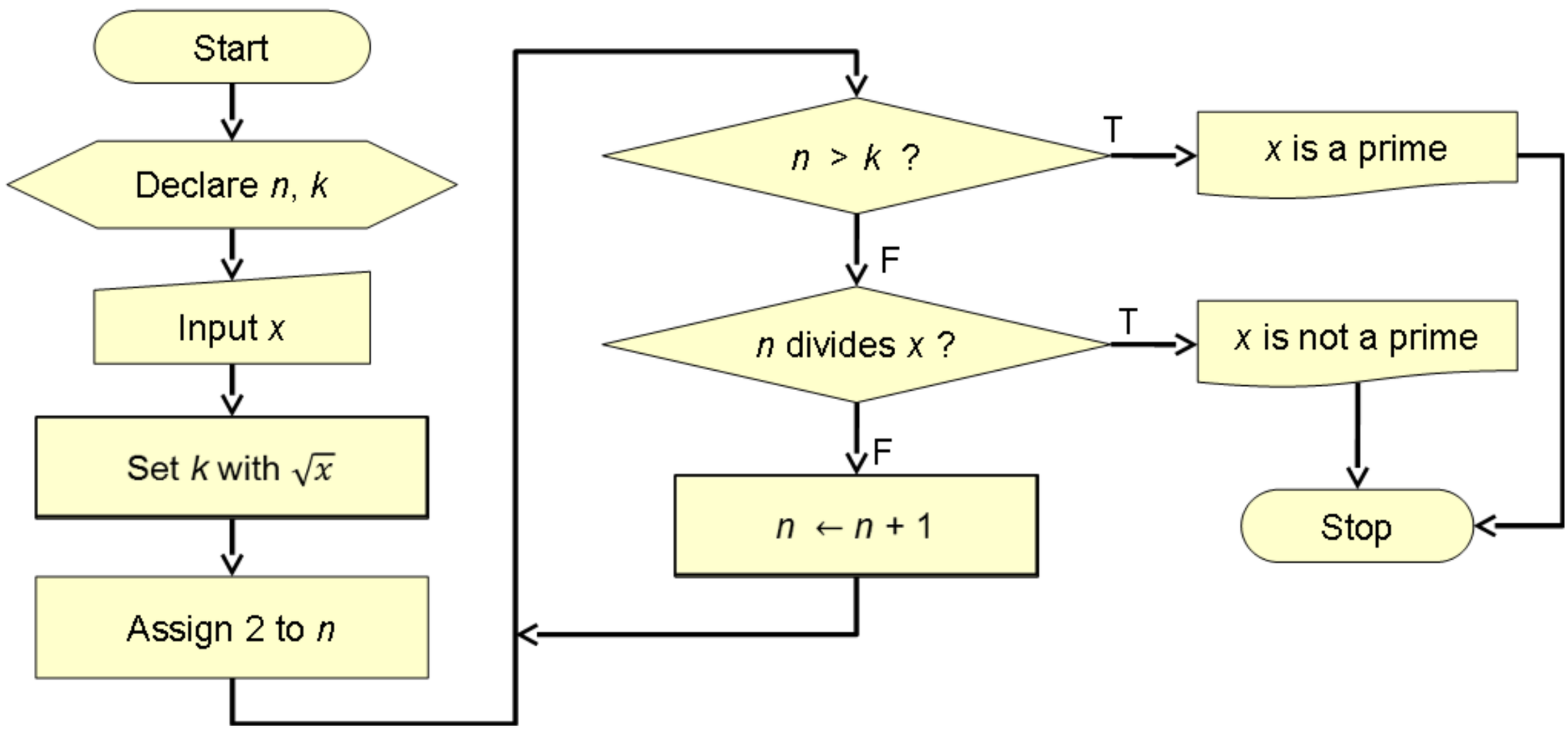
- Recall what a prime number is.

**A prime number is a natural number that has exactly two distinct natural number divisors: 1 and itself**

# Problem Solving with Computer

- A straightforward (simple ?) version:
  - Input x
  - Check if 2 divides x. If True, x is not a prime
  - Check if 3 divides x. If True, x is not a prime
  - ...
  - Check if k divdes x. If True, x is not a prime.
  - If False, x is a prime.
- What is k?
  - the largest natural number which is at most
  - Think about why??

# Problem Solving with Computer



# Programming

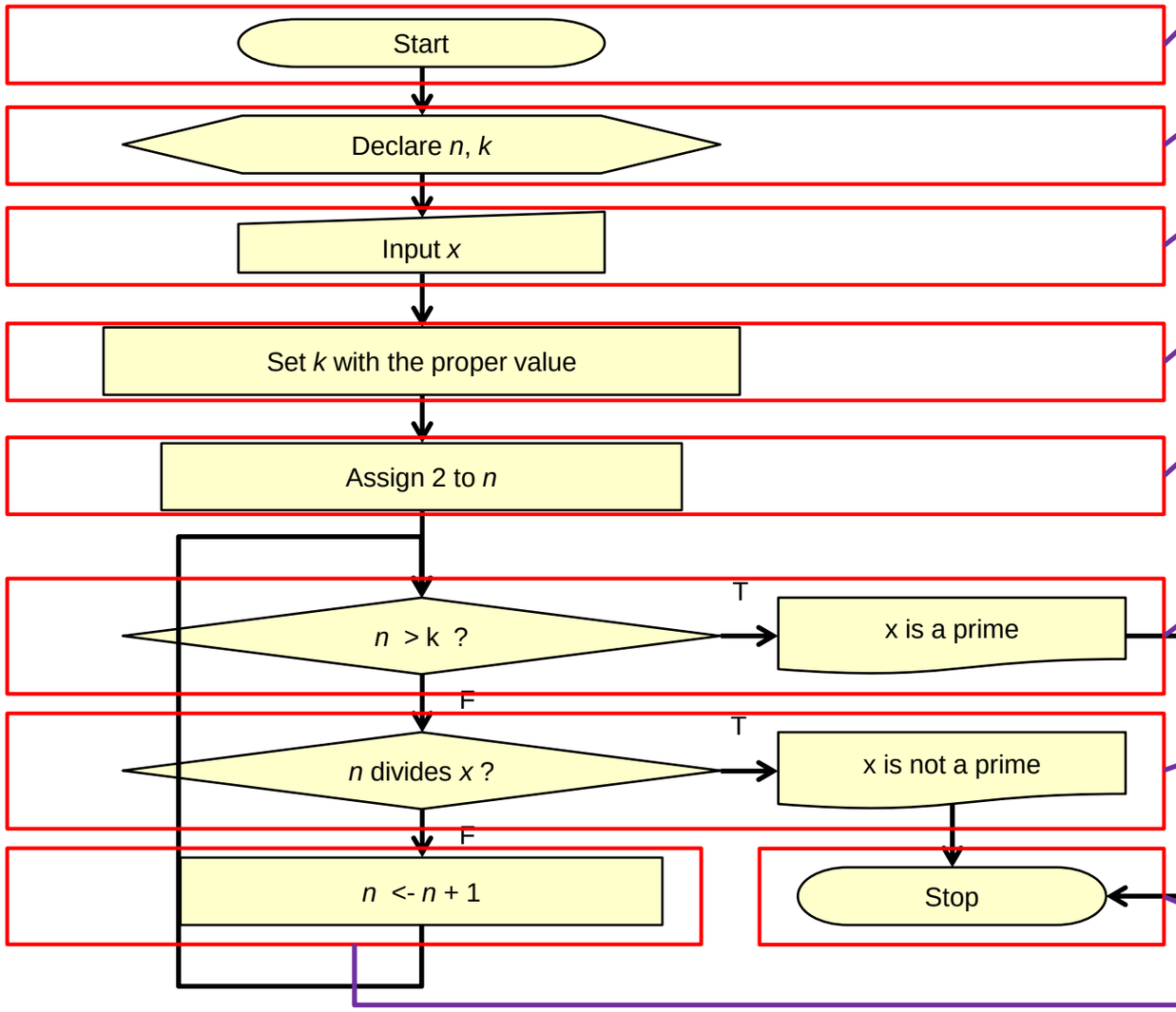
- Flow chart:
  - a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows.
- Why flow chart and algorithms?
  - This diagrammatic representation can give a step-by-step solution to a given problem.
  - Flowcharts are used in analyzing, designing, documenting or managing a process or program

# Programs

**A sequence of instructions written to perform a specified task for a computer**

# Program

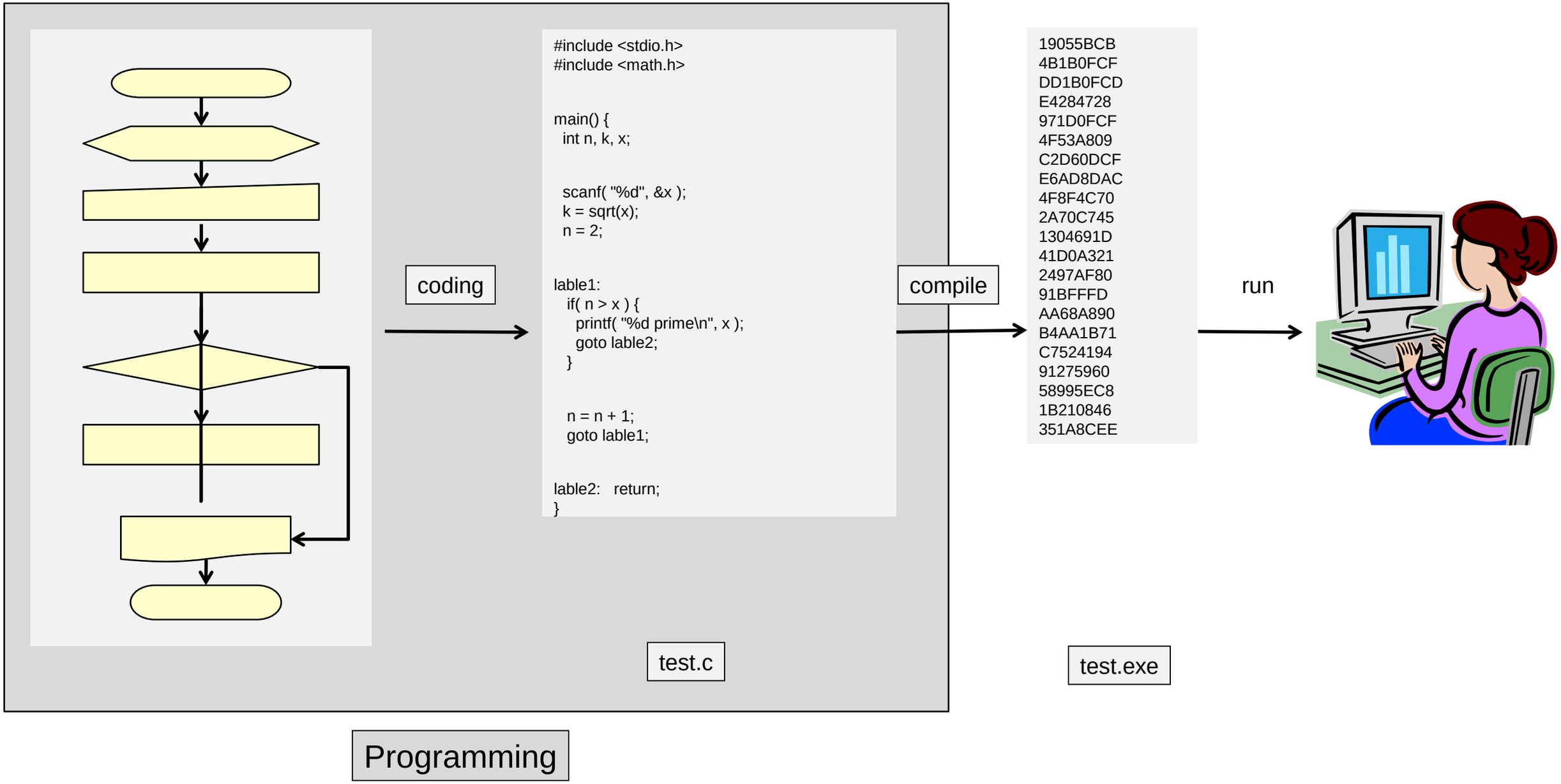
- A list of instructions




```
#include <stdio.h>
#include <math.h>

main() {

    int n, k, x;

    scanf( "%d", &x );

    k = sqrt(x);

    n = 2;

lable1:
    if( n > k ) {
        printf( "%d is a prime\n", x );
        goto lable2;
    }

    if( x % n == 0 ) {
        printf( "%d is not a prime\n", x );
        goto lable2;
    }

    n = n + 1;
    goto lable1;

lable2:
    return;
}
```

# Programming

- What are programs for ?



# Programs

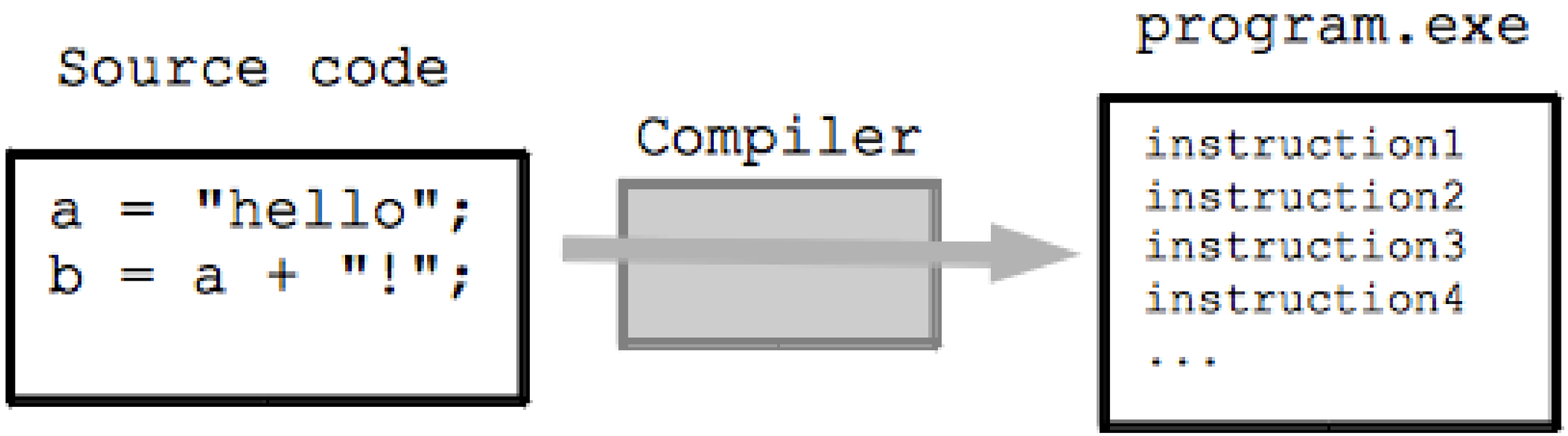
**Coding is to translate an algorithm into a sequence of instructions written to perform a specified task for a computer**

- Programming Language?
  - an artificial language designed to express computations that can be performed by a computer
  - C, C++, Java, Perl, Basic, Pascal, Fortran, COBOL, …

## Compiler

- "Compiler" looks at the source code
- Compiler translates the source code into a large number of machine code instructions
  - e.g. Firefox -- written in C++
  - Compiler takes in Firefox C++ source code, produces Firefox.exe
- The end user does not need to the source code or the compiler. Distribute the program.exe file in working form
- Does not work backwards -- having the .exe, you cannot recover the source code (well)

# Visual Representation of Compiler




The **Compiler** for the C++ language, (1) reads that C++ code and (2) translates and expands it to a larger sequence of the machine code instructions to implement the sequence of actions specified by the C++ code

# History of C Language

- Developed at Bell Lab., 1972 for system-level programming.
- Used for implementing Unix OS

BCPL (Basic Combined Programming Language)
➔ B language (Ken Thompson)
- ➔ C language

# Advantages of C Language

- Efficient
  - Developed for low-level (machine-level) execution
- Portability
  - Applicable to virtually all platforms from PCs to Supercomputers
- Powerful
  - Provides various data types and operators
- Flexibility
  - Applicable from system-level to application-level programming
- Many Standard Libraries
  - Input/Output, String handling, Storage allocation, …

# Disadvantages of C Language

- Error Prone
- Difficult to detect errors resulting from its flexibility
  - 
- Difficulty
  - Difficult to understand and modify it due to many functionalities

# Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdio.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**Comment**

-Ignored by compiler

-Used by programmer to explain the code, or include other information about the code

One line comment

- // One line comment

Multi-line comment

- /* Multi-line comment
- Multi-line comment
- Multi-line comment */

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdio.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**Lines begin with #**

-Preprocessor directive

-Used for including external librares

stdio.h is the Standard Input/Output library

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdio.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

### The main function

-Every program has a main function
-Execution of the program begins here
-int: this function returns an integer as a return value
-void: this function takes no argument as input

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdio.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**{**

-Every function begins with a brace “{“, and ends with a “}”
-Braces are also used to group statement together

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdio.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**printf**

-A function in stdio.h
-Print words on screen

**Except Characters**

-Starts with “\”
-Used to display “special” characters
e.g.
\n – new line character
\t – tab
\\ – slash
\a – alert

```c
// Fig. 2.3: fig02_03.c
// Printing on one line with two printf statements.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   printf( "Welcome " );
   printf( "to C!\n" );
} // end function main
```

```
Welcome to C!
```

**Fig. 2.3** | Printing on one line with two `printf` statements.

```c
// Fig. 2.4: fig02_04.c
// Printing multiple lines with a single printf.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   printf( "Welcome\nto\nC!\n" );
} // end function main
```

```
Welcome
to
C!
```

**Fig. 2.4** | Printing multiple lines with a single `printf`.

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdafx.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**;**

-Every line of code ends with a “;”
-Work as a “full-stop” in English
-Tell the compiler this is the end of the line of code
-If you don’t include “;”, multiple lines will be treated as one line of code

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdafx.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**return 0;**

-The integer value zero is returned by the main function at the end

-Non-zero value are used to tell the OS that main() has been unsuccessful

## Your First Program: HelloWorld.c

```
// The HelloWorld Program
#include “stdafx.h”

int main(void)
{
  printf(“Hello, World!\n”);
  return 0;
}
```

**Empty spaces**

-Note that empty spaces such as new line, tab, space were not read by the compiler

**Indentation**

-Most programmer uses tab or space to indent the code
-Increase readability for the code
-Easier to debug

# Syntax and Semantics

- Programming languages (called “artificial languages”) are languages just as “natural languages” such as English and Korean. ***Syntax*** and ***Semantics*** are important concepts that apply to all languages.

# Syntax

- The **syntax** of a language is a set of characters and the acceptable sequences (arrangements) of those characters.
- English, for example, includes the letters of the alphabet, punctuation, and properly spelled words and sentences. The following is a syntactically correct sentence in English,

**“what time is it now?”**

- The following, however, is not syntactically correct,

**“what tyme is it now?”**

- The sequence of letters “**tyme**” is not a word in the English language.

# Semantics

- The **semantics** of a language is the meaning associated with each syntactically correct sequence of characters.
- Consider the following sentence:

  **"A silent noise drinks stones."**

  This sentence is syntactically correct, but has no meaning.
- Thus, it is ***semantically*** *incorrect.*
- Every language has its own **syntax** and **semantics**.

# Semantic Error

In contrast, **semantic errors** are errors in **program logic**.

Such errors cannot be automatically detected, because translators cannot understand the original intention.

For example, if a program computes the average of five numbers ;

( 5 + 6 + 3 + 8 + 1) / **4**

The divisor should be **5** and not **4**, but translators do not understand what a programmer is meant to do.

**=> It is up to the programmer to detect such errors.**

Program debugging is not a trivial task, and consumes much of the time of program developments.

# Debug: Finding out error in code

**Program debugging** is the process of finding and correcting errors

- (**“bugs”**) in a computer program.

**Syntax Error**

- **Syntax errors** are caused by invalid syntax. (for example, entering **prnt** instead of **print**).

- Since a translator (compiler or interpreter) can detect syntax errors (e.g. prnt), translators terminate and indicate where the error occurred.

# Q&A?