





CECS130-Introduction to Programming Languages

Lecture slides for this and upcoming lectures are based on the slides by Dr. Imam, Professor Berenbaum and those provided by the book publishers and internet sources.

Dr. Roman Yampolskiy

Instructor: **Dr. Roman V. Yampolskiy**

- Office: Duthie Center, Room 215.
-  (502) 852-3624
-  roman.yampolskiy@louisville.edu
- **Office Hours:** (subject to change)
 - Monday: 2:30 – 3:30PM
 - Wednesday: 2:30-3:30PM
- **Appointments** (at least 24 hour notice required)



Lectures and Labs

- Lectures:
 - Monday and Wednesday: 11:00 – 11:50PM
 - (Duthie Center 117)
- Lab 01:
 - Tuesday 11:00 – 12:50PM
 - (Duthie Center 117)
- Lab 02:
 - Thursday 11:00 – 12:50PM
 - (Duthie Center 117)

Lab Assistants:

- **Name:** Elaheh Arabmakki; **Office:** Duthie Center 241
- **Phone:** (502) 852-3624; **Email:** [e0arab01 @louisville.edu](mailto:e0arab01@louisville.edu)
- **Office Hours:** T, Th 4-5:30PM

Introduction to Programming CECS130

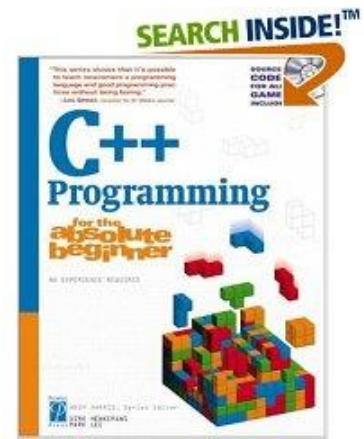
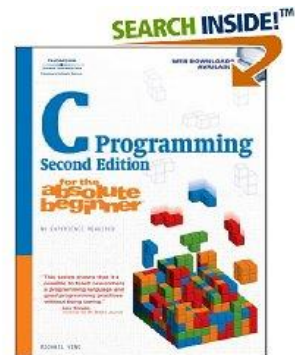
- Prerequisites: Basic knowledge of computers (mouse, keyboard, etc.).
- Course Description: Introduction to programming languages including laboratory exercises on the writing and compiling of computer programs in C and C++. This course will cover topics pertaining to program life-cycle, modularity, operators, expressions, flow control, functions, scope, memory management, structures and data files, Object-Oriented Program design and development.

Course Objectives:

- Understand the syntax of the C and C++ language.
- Use the language programming environment, including editors, compilers, linkers and create an executable program written in C or C++.
- Design and implement problem solving algorithms.
- Apply sound structured programming principles.
- Acquire a solid foundation in Object-Oriented design and development

Textbooks: Two Books are required:

- *C Programming for the Absolute Beginner, Second Edition* by Michael Vine
ISBN 13: 978-1-59863-480-8 © 2008
ISBN 10: 1-59863-480-1
- *C++ Programming for the Absolute Beginner* by Dirk Henkemans, Mark Lee
ISBN 13: 978-1-931841-43-6 © 2002
ISBN 10: 1-931841-43-8
Course Technology.
CENGAGE Learning (Formerly Thomson Learning)



Labs:

- When you register for this course, you are also registering for the lab session for this course. Students are **REQUIRED** to attend the lab section that they are registered for. Assignments involve writing programs assigned from the textbook or supplementary materials. You should expect to spend some time working on these exercises outside of your scheduled lab time. Assignments are due by the end of the **NEXT** lab. So for example: assignment 2 is due by the end of lab 3.
- **Late lab = 0**

Grading:

A+

- Laboratory/ Programming Assignments: 50%
- Two Midterm Examinations: 30%
- Final Comprehensive Examination: 20%
- The grading scale is as follows:
 - 100 – 90 A
 - 89.99 – 80 B
 - 79.99 – 70 C
 - 69.99 – 60 D
 - 59.99 – 0 F

Exams:



- Two mid-term exams and one final exam will be administered. The exams will be based mainly on the laboratory and project work you have completed, as well as the reading assignments and class lectures. The exams are closed book, closed notes.
- Exam make-ups: Prior notice must be given to your instructor. No make-ups will be granted unless satisfactory documentation is produced to show an extenuating circumstance.



Grading and Attendance

- Grading questions: If you have a question about a grade, you should see your instructor within one week of the day the graded work is returned to you. You lose the right to re-grading after that.
- Incompletes: Incomplete grades (I) are granted very rarely and only under extenuating circumstances.
- Absences: A good record of class and lab attendance and participation may help your final grade in borderline cases. Making up material from missed classes is your responsibility.

Academic dishonesty:




- Students are expected to do their own work.
- **Copying is strictly forbidden.**
- Academic dishonesty is defined in the Code of Student Rights and Responsibilities.
- It is the student's responsibility to become familiar with the Code.
- Allegations of academic dishonesty are handled in accordance with the Procedures for Dealing with Breaches of Academic Integrity.
- Copies of the Code are available in the Speed School Academic Affairs Office and departmental offices.

Weekly Schedule:


Week	Lecture Date	Topic	Reading	Lab Date
1	Monday, 8/26	Getting Started	C1	Tuesday, 9/27
	Wednesday, 8/28	Data Types	C2	Thursday, 9/29
2	Monday, 9/2	Labor Day		Tuesday, 9/3
	Wednesday, 9/4	Conditions	C3	Thursday, 9/5
3	Monday, 9/9	Loops	C4	Tuesday, 9/10
	Wednesday, 9/11	Functions	C5	Thursday, 9/12
4	Monday, 9/16	Arrays	C6	Tuesday, 9/17
	Wednesday, 9/18	Pointers	C7	Thursday, 9/19
5	Monday, 9/23	Strings	C8	Tuesday, 9/24
	Wednesday, 9/25	Data Structures	C9	Thursday, 9/26
6	Monday, 9/30	Dynamic Memory	C10	Tuesday, 10/1
	Wednesday, 10/2	Midterm 1 Review		Thursday, 10/3
7	Monday, 10/7	Mid-Term Break		Mid-Term Break
	Wednesday, 10/9	Midterm 1		Tuesday, 10/10
8	Monday, 10/14	Midterm 1 Results		Thursday, 10/15
	Wednesday, 10/16	Files	C11	Tuesday, 10/17
9	Monday, 10/21	Input/output	CPP1	Thursday, 10/22
	Wednesday, 10/23	Data, Control, Fun.	CPP2-4	Tuesday, 10/24
10	Monday, 10/28 W	OOP	CPP5	Thursday, 10/29
	Wednesday, 10/30	Namespaces	CPP6-7	Tuesday, 10/31
11	Monday, 11/4	Inheritance	CPP8	Thursday, 11/5
	Wednesday, 11/6	Templates	CPP9	Tuesday, 11/7
12	Monday, 11/11	Midterm 2 Review		Thursday, 11/12
	Wednesday, 11/13	Midterm 2		Tuesday, 11/14
13	Monday, 11/18	Midterm 2 Results		Thursday, 11/19
	Wednesday, 11/20	Error Handling	CPP11	Tuesday, 11/9
14	Monday, 11/25	Recursion		Thursday, 11/21
	Wednesday, 11/27	Thanksgiving		Thanksgiving
15	Monday, 12/2	Student ppt		Thursday, 12/3
	Wednesday, 12/4	Student ppt		Tuesday, 12/5
16	Monday, 12/9	Student ppt		
	Final Exam:	Monday December 16 11:30 - 2:00PM	DC117	

Blackboard



 Roman Yampolskiy My Places Home Logout

CECS 130-01: INTRO TO PROG LANG-Spring 2011 Announcements



CECS 130-01:
INTRO TO PROG
LANG-Spring
2011

Start Here

Syllabus

Faculty Information

Course Description

Announcements

Course Documents

Assignments

Communication


Homepages

My Grades

Tools

Instructors Start Here

Tegrity Classes



Announcements

Create Announcement

No Announcements found.

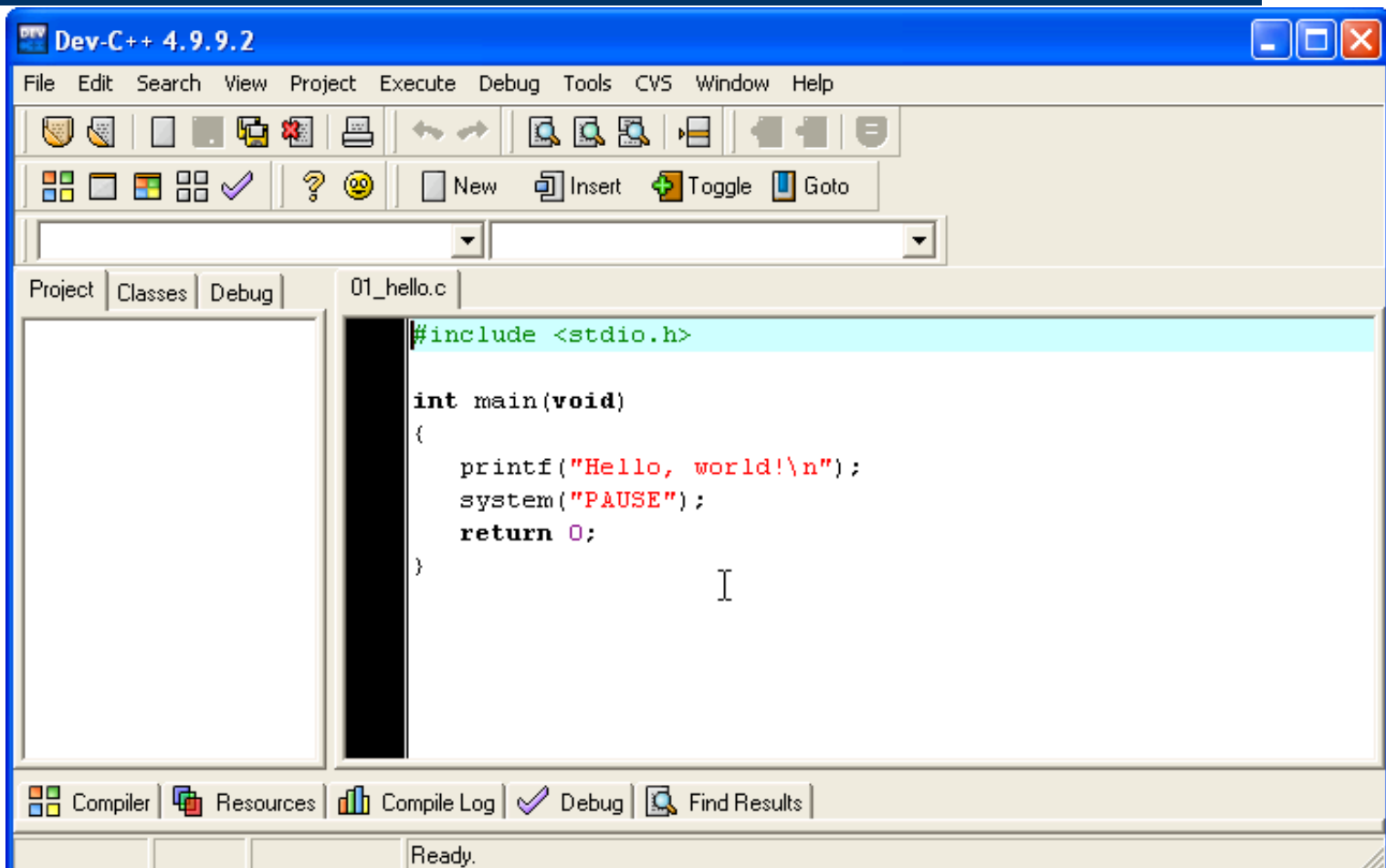


Programming and Preparation

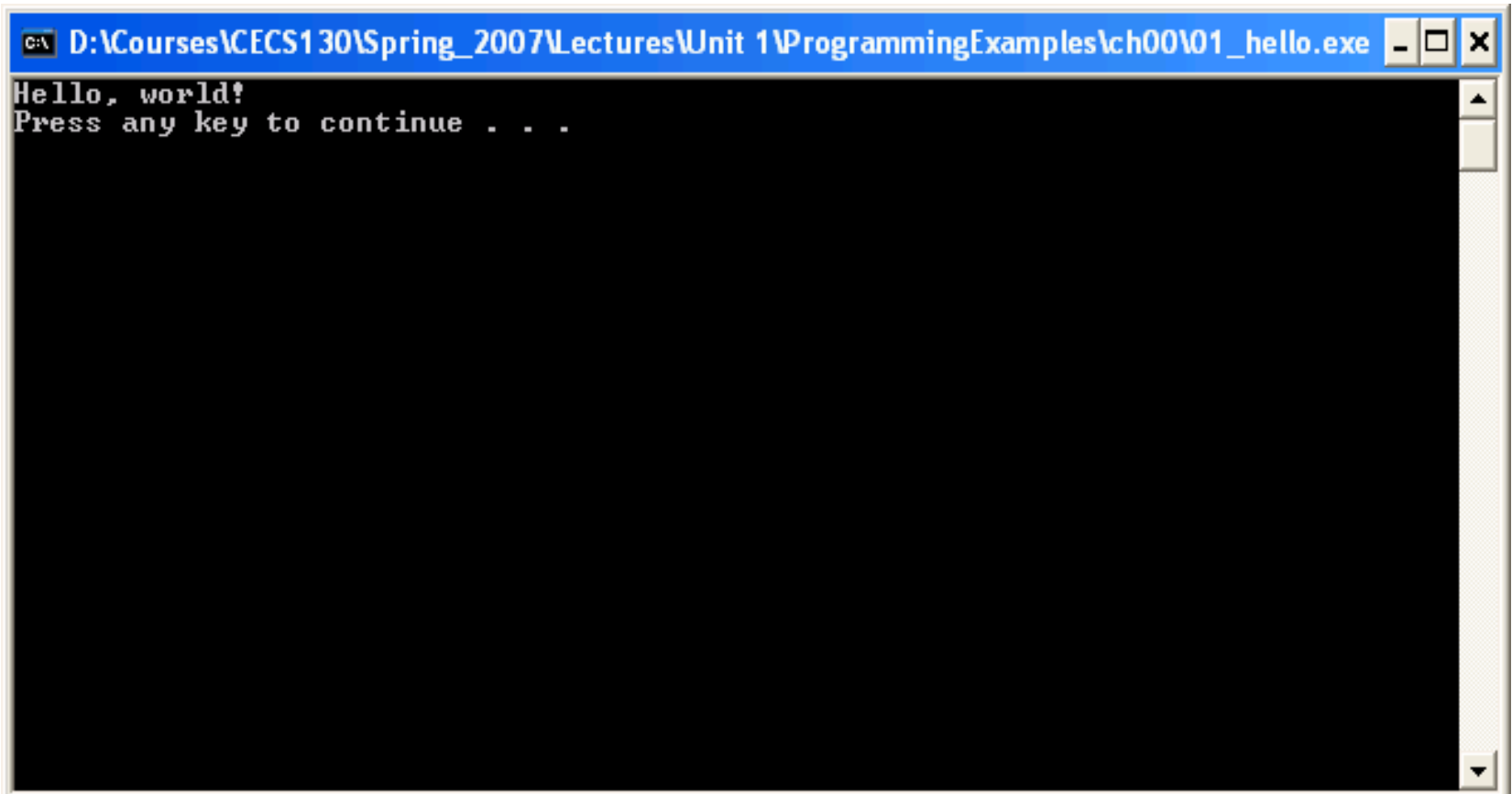
- **Operating System** is a collection of special programs running on each computer and facilitating the use of that computer.
 - Examples are Windows VISTA, XP, Linux, UNIX, MAC OS X.
- Some operating systems such as Linux/UNIX come with a C compiler such as cc or gcc.
- Other operating systems such as Windows require the installation of a purchased c compiler such as the one that comes with MS Visual Studio or an Open Source one such as Bloodshed's Dev-C++



Software- Dev-C++ (Bloodshed)



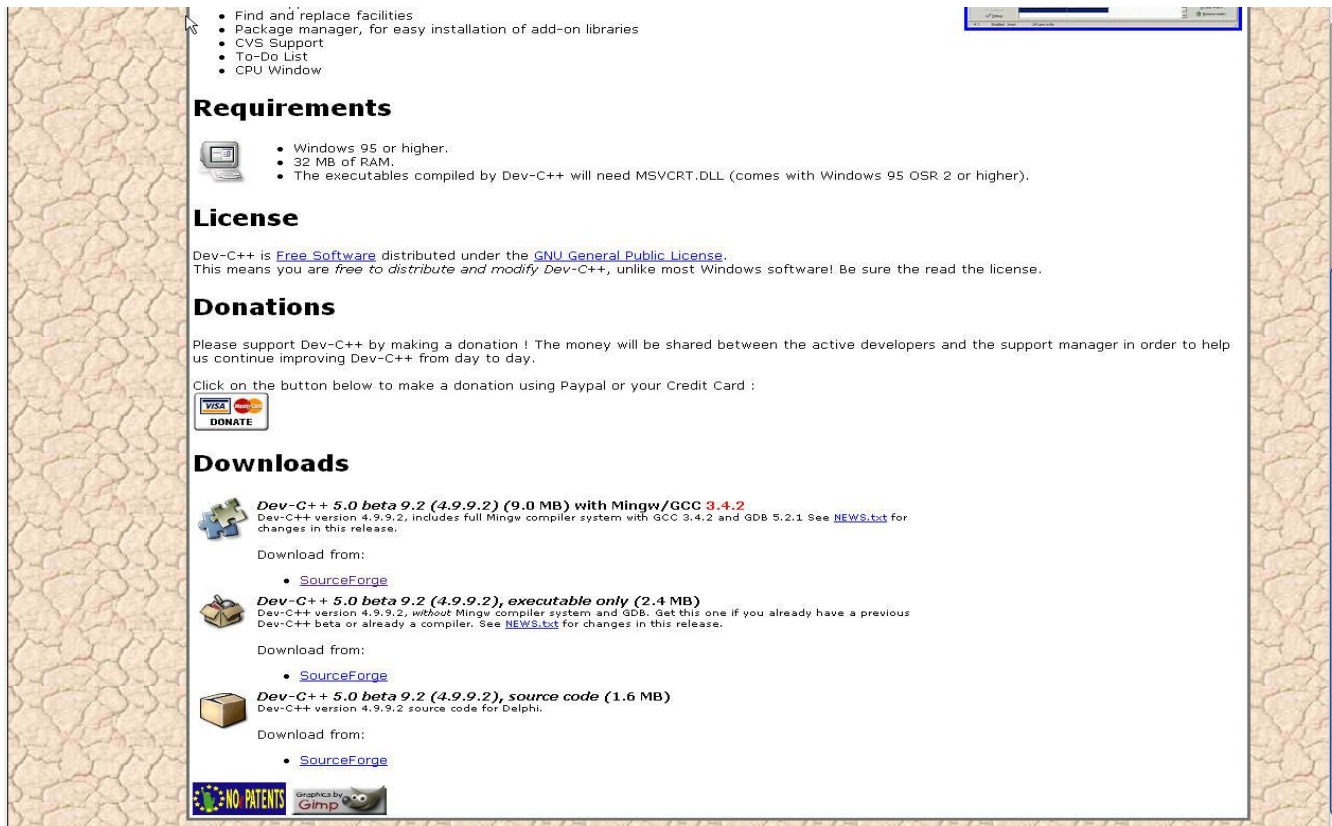
Output Window



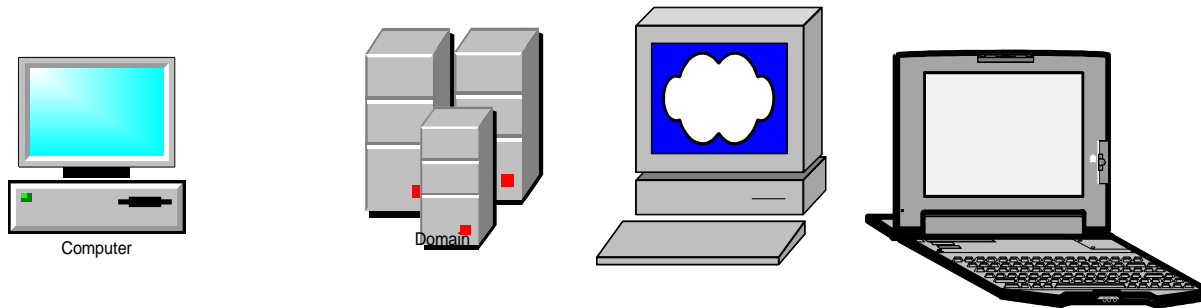
```
C:\ D:\Courses\CECS130\Spring_2007\Lectures\Unit 1\ProgrammingExamples\ch00\01_hello.exe
Hello, world!
Press any key to continue . . .
```

Downloading and Installing Dev-C++

- <http://www.bloodshed.net/dev/devcpp.html>

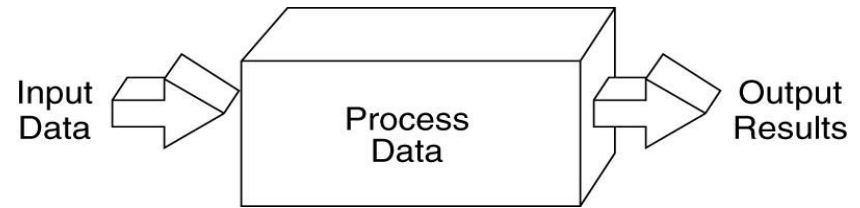


What is a computer?

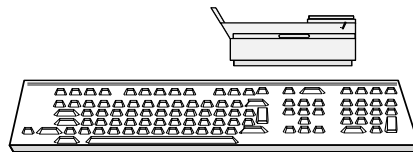


But what is it really? Technically speaking?

Computer



- Data processing device
- Takes input from environment
- Generates output
- May store intermediate results

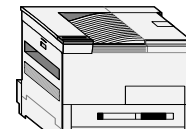


Input Devices



CPU

Processor



Output Devices



Computer

- Dumb device.
- Only does what it is told.
- So dumb, that it only understands extremely explicit instructions.
- We must use a special language: ***Programming Language***.
- Programming Language – Language in which a programmer can explain to the computer what he/she needs it to do.

Programming Language



- Computers are complex networks of switches.
- Only understand sequences of 0's and 1's, where 0 represents the off state of the switch and 1 represents the on state of the switch.
- Long patterns of 0's and 1's represent instructions and data.
- This is a very complicated way of communicating with the computer.
- A programmer might spend hours programming the machine using 0's and 1's to perform something as simple as adding two numbers.

Programming Language

- As systems became more complex, so have the means of communicating with them.
- Flipping switches evolved into pointing with the mouse and clicking.
- Programming evolved from entering long patterns of 0's and 1's using toggle switches, to writing what needs to be done in an “English-Like” language.

Creating a Computer Program

- A program is similar to a recipe in a cook book ~ a step-by-step set of instructions for the computer to follow.
- A program is written in a programming language.
- So how exactly do we do that?
- Three steps are involved.

Creating a Computer Program

1. Write the program in programming language using a text editor. Save the program in a file. This file contains what's known as the **source code**.
2. Compile the program using a special program known as a **compiler**. A compiler converts a source code into a sequence of 0's and 1's.
3. A file containing 0's and 1's is generated. This file is referred to as the **executable file**.
"Run" the executable file.

Natural VS. Programming Language

- Programming languages are like spoken languages.
- They have syntax, semantics, vocabularies, and grammars.
- Programming languages are developed by people for a specific purpose.
- They are more like professional jargons.

Which Language to Use?

- Each language was designed to allow the programmer to explain a very specific task to the computer in the most effective manner.
- We can be much more effective creating a network based program using Java than we would have been using Fortran.
- However, for complex number crunching, huge matrix manipulations, and other such tasks, Matlab is far more effective than Java.

Survey of Programming Languages

- **FORTRAN** (**FOR**mula **TRAN**slation) was introduced in 1957 and was used to translate formulas into computer readable format.
- **COBOL** (**CO**mmon **B**usiness **O**riented **L**anguage) was introduced in 1960 and was used for translating business applications into computer readable format.
- **BASIC** (**B**eginners **A**ll-purpose **S**ymbolic **I**nstructions **C**ode) was developed in 1960 at Dartmouth College as a straight forward language that is easy to understand.
- **Pascal** was developed in 1970 for the purpose of introducing a modular structured language and became mainly a language used for educational purposes.
- **C** was developed in the 1970s at Bell Labs. It has its roots in a language called **B**. C has an extensive set of capabilities and gained wide acceptance as the “professional programming language” and is currently maintained by the American National Standards Institute (ANSI).

C and C++

- We are going to be using C and C++.
- C++ is a “universal language” and an extension of C.
- It can be used for just about any task, however, a “specialized” language would probably be more effective.
- It contains the majority of paradigms present in other languages.
- If you become “good” in C++, you will have very little trouble picking-up other languages, should you ever need to do so.
- This may not necessarily be the case if you start learning to program in a highly specialized language.

What Does a Program Look Like?

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
    printf("\nC you later\n");  
  
}
```

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
    printf("\nC you later\n");  
}
```

Let's Dissect the Program

- Comments
 - Allow programmer to annotate what he/she wrote
 - Explain what each section of the code does
 - Help with management of long programs
 - Help remembering what was done after being away for a while
 - Help other programmers (colleagues) understand the code

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
    printf("\nC you later\n");  
}
```

Let's Dissect the Program

Two ways to write comments

`/* ... */`

- Everything between these is ignored by the compiler.
- Useful when writing large chunks of text in the code to describe or annotate something.

`// ...`

- C++ specific.
- Everything past `//` is ignored by the compiler.
- Useful when placing comments on the same line as code.
- Temporarily removing lines of code from the program.
- One-line annotations.

Let's Dissect the Program

- “Larger” comment:

```
/*  
    This is a simple C Program.  
  
    Call this file Sample.c  
*/
```

- “Smaller” comment:

```
// A C++ program begins at main().
```

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
    printf("\nC you later\n");  
}
```

Let's Dissect the Program

`#include <stdio.h>`

- C defines headers that contain information needed by the program.
- `<stdio.h>` supports Input/Output system of C (I/O)
- `<stdio.h>` provided with the compiler
- We use `#include` statement to include headers
- We will learn towards the end of the course what the headers are, why we need them, and how to write them.

Let's Dissect the Program

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
    printf("\nC you later\n");  
}
```

`main()`

- This is the entry point into the program.
- This is what's known as a *function* (more on functions later)
- Function defines a set of actions to be performed.
- The function body is defined by `{ }`. The code between the curly brackets is the function body.
- Every C program has the `main()` function.

Let's Dissect the Program

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
    printf("\nC you later\n");  
}
```

```
printf("\nC you later\n");
```

- Statement to output something to the **console**.
- The console is the DOS window. We will write **console applications** in this course.
- **\n** tells the computer to start a new line
- In this case, *C you later* will be displayed on the screen.
- The message "*C you later*" is called a **string**. Strings are displayed character by character to the screen. **Whatever is included in the double-quotes is displayed on the screen literally.**

Escape Sequence Description

- **Used to format output**

- **\'** **Single quote**
- **\"** **Double quote**
- **** **Backslash**
- **\n** **Newline**
- **\r** **Carriage return**
- **\t** **Horizontal tab**

```
/*  
This is a simple C Program.  
*/  
// This is a comment  
  
#include <stdio.h>  
  
main()  
{  
  
printf("\nC you later\n");  
}
```

Let's Dissect the program

- Note the semicolon (;) on the end of the line.
- This indicates the end of the statement.
- Another name for the semicolon is statement terminator.



Programming Style

Both programs below will produce identical results, but if you are to work on one of these to fix a bug which one would you rather work with?

```
#include <stdio.h>
main()
{
    printf("Hello world\n");
    system("PAUSE"); |
} /* eof main */
```

```
#include <stdio.h>
main() { printf("Hello world\n"); system("PAUSE"); }
```

Algorithms

Step by step instructions for implementing method three will be as follows:

- 1) Set $n = 100$
- 2) Set $a = 1$
- 3) Set $b = 100$
- 4) Set $\text{sum} = n \cdot (a+b)/2$

- A step-by-step sequence of instructions that describes how to perform a computation. The programmer must clearly understand what the outcome of a procedure should be and how to achieve it.

Method 1. Columns: Arrange the numbers from 1 to 100 in a column and add them:

1
2
3
4
⋮
⋮
98
99
100
5050

Method 2. Groups: Arrange the numbers in convenient groups that sum to 100. Multiply the number of groups by 100 and add in any unused numbers:

0	100	100
1	99	100
2	98	100
3	97	100
⋮	⋮	⋮
⋮	⋮	⋮
49	51	100
50	0	50

50 groups
(50 ■ 100)
50
5050

One unused number

Method 3. Formula: Use the formula

$$\text{Sum} = \frac{n(a+b)}{2}$$

where

n number of terms to be added (100)
 a first number to be added (1)
 b last number to be added (100)

$$\text{Sum} = \frac{100(1+100)}{2} = 5050$$

From Algorithms to Programs

- Converting an algorithm into a program is called **coding**.
- Coding is done using a programming language such as C which is easily understood by programmers but not necessarily by computers.
- Such language is normally called **high level language**.
- The part of the program written in the high level language is called **source code** and is saved in what we call a **source file**.
- Instructions that consist of 1s and 0s and are understood by the computer are called **machine instructions** or collectively as **machine language**.

Translator-Interpreter

- To convert from source code to machine instructions we must use a translator.
 - If we chose to use a translator that translates one instruction at a time and executes the corresponding step immediately, then we are using an **interpreter**. Languages that allow for such translation are called **interpreted languages**.

Translator-Compiler

- If we chose to translate the entire algorithm at once and then execute it at a later stage, then we are said to be using a **compiled language**.
- Translators used with compiled languages are called **compilers**.

Linker

- Compiled code is machine code but it is not stand alone code.
- To make it so we must append additional code provided by the designer/vendor of the compiler. Such a process is called **linking** and is performed by a **linker** program.
- The result of compiling and linking is a stand alone program (executable) that can be run on the computer for which it was designed.

Review

- **Computer Program:**

A sequence of instructions used to operate a computer to produce a desired result.

- **Programming:**

The Process (more than one step) of writing these instructions in a language that the computer can respond to and that other programmers can understand.

- **Programming Language:**

The set of instructions that can be used to construct a program.

The End!

