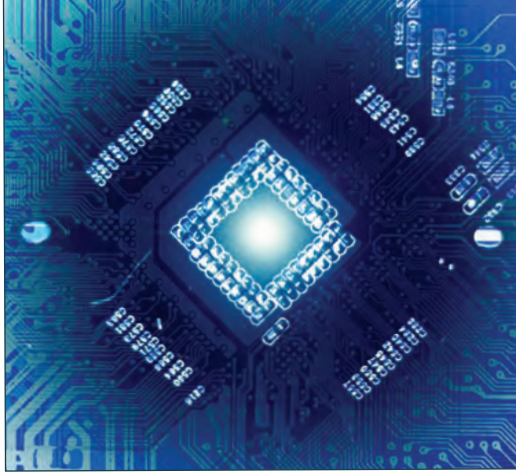
CHAPTER 1



Introduction

Objectives

After completing this chapter, the student should understand and be familiar with the following:

- Some terminology of computer hardware
- Some terminology of computer software
- Algorithm
- Flowchart
- Programming languages
- C++ and Object Oriented Programming
- Programming documentation policy
- Starting a program in C++ and understanding key terms
    1. Comment
    2. #include directive
    3. "using namespace std;" statement
    4. int
    5. main(void)
    6. cout <<" ";
    7. system ("cls");
    8. system ("Pause");
    9. " \n " as a carriage return
    10. " \t " as a Tab key
    11. open and close curly brackets ({ })

1.1 Introduction

The purpose of writing this book is to share my personal experience and knowledge in learning and teaching programming—and to make the programming as easy as possible. It is written in a simple way with a simple language so that even students without a background in programming can follow and learn. All programs in the chapters are compiled and tested with no errors. All programs in this book are written in C++ language using Microsoft's Visual Studio 2010. Microsoft is a registered trademark for Microsoft Corporation in the United States of America and other countries. The output of each program is also displayed so that students can see the result. I always advise my students that knowledge is power—it can neither be bought, nor can it be injected. Learning how to design a program requires time and patience.

Programming is not only a design; it is also an art that you must enjoy. Do not be afraid to learn how to design a project and implement it using any programming language. As a student, try to understand the problem and then design your program on paper. Once you are done with the design, implement it with your programming language of choice. Then trace your program for all possible inputs, making sure it works for all cases. The more time you spend on the design, implementation, and testing on the paper, the less time you spend on the computer. The more you read and the more you practice, the easier it gets for you to approach the problem and solve it.

> Constructivist teachers encourage students to constantly assess how the activity is helping them gain understanding. By questioning themselves and their strategies, students in the constructivist classroom ideally become "expert learners."[1]

Both teaching and learning should be constructive. The constructive way of teaching is student-centered based, as opposed to teacher-centered. It requires active engagement of students in the classroom as part of their participation in learning. Constructive learning is the ability to start a project and finish it. It is also the ability to work with others as a team—regardless of personal conflicts. Furthermore, constructive learning is the ability to learn without having any fear and without complaining. It is the ability to go through the process recursively. To be successful you must enjoy learning and have the passion to read and learn.

Knowledge is power, and it can neither be bought nor injected. Human knowledge is limited, however. I don't believe anybody's knowledge is perfect or complete. The more we learn, the more we realize that we know less. We can learn from each other if we are willing. The sky would (or could be) the limit for the success. Throughout my 27-year-teaching career as Associate Professor of Computer Science at Texas Southern University, I have learned that having the following behaviors would help anybody to be a better person in life:

1. Work hard.
2. Be patient.
3. Take care of your health.
4. Watch your diet.
5. Exercise regularly.
6. Read regularly.
7. Meditate.
8. Believe in yourself.
9. Be open-minded.
10. Work with others.
11. Share your knowledge and information with others.
12. Be honest.
13. Think positive.
14. Be happy.
15. Never settle for less.
16. Aim high for success

1. Wnet education. "What is Constructivism?" *Constructivism as a paradigm for teaching...-thirteen*, accessed October 15, 2014. http://www.thirteen.org/edonline/concept2class/constructivism/

17. Be generous.
18. Respect others and be respected.
19. Be humble,
20. Strive to learn.
21. Help others.
22. Have peace of mind.
23. Accept responsibility for your failures.
24. Be proactive
25. Travel to learn about other cultures.
26. Always look at both sides of the coin.
27. Avoid retaliation.
28. Avoid segregation.
29. Avoid discrimination.
30. Avoid any kind of violence.
31. Be ethical.
32. Be disciplined.
33. Be on time.
34. Be ahead of any schedule you set.
35. Do not make decision under stress.
36. Do not make any decision when you are either under confident or over confident. Think deeply.
37. Never give up hope when you feel under confident.
38. Do not get too comfortable when you feel overconfident.
39. Try to explore life.
40. You have two ears and one tongue. Try to listen more than you talk.
41. Think before you open your mouth. There will always be time to say what you are supposed to say.
42. Be proud of what you are.
43. Be proud of who you are.
44. Know your roots.
45. Continue what you started until you reach your goal.
46. Have the ability to forgive, but not to forget.
47. More important than any other power is the power of reputation in the society.
48. Keep your integrity high.
49. Be a leader.
50. No pain, no gain.

1.2 Computer Hardware

Key terms:

A **program** is a list of instructions.

A **computer** is a dummy machine by itself. The software makes the computer a smart machine.

A **programmer** is a person who writes the program that makes the dummy machine smart.

Computer systems are composed of computer hardware and software.

Computer hardware is the physical part of a computer system.

Components of computer hardware include the following:

1. central processing unit (CPU)
2. main memory
3. secondary storage
4. input devices
5. output devices

1.2.1 CPU

The CPU is the brain of the computer, which fetches or gets the instruction from the main memory and then processes it. It consists of Arithmetic Logic Unit (ALU) and Control Unit (CU).

- ALU is used to do both mathematical and logical operations.
- The control unit (CU) coordinates all activities of the computer, analogous to a traffic light on a four-way street.

1.2.2 Memory

Main memory or **random access memory** is used to store information.

Bit: Information stored in memory are string of 0's and 1's. Each 0 or 1 is called a **bit**.

Byte: Eight bits equals one **byte**, which represent one character.

1.2.3 Secondary Storage

Secondary storage: Due to the limited space of main memory, another type of storage is needed to store large amount of information. This type of information can be stored in a hard drive or any other external device and is called **secondary storage**.

1.2.4 Communication Devices

Communication devices: Input / Output devices are also called **communication devices**.

Input devices: The information we give to the computer is called **input**. Some of the input devices are the keyboard and the mouse.

Output devices provide output to the user.

1.3 Computer Software

Computer software is a collection of programs.

> "An **operating system** is a program that manages the computer hardware. It also provides a basis for application program and acts as an intermediary between the computer user and the computer hardware"[2]

An operating system allows a user to communicate with the computer. The operating system is a **program** or a list of instructions. **Software** is a collection of programs.

A **software engineer** is a programmer who designs, implements, compiles, tests, verifies, validates, and maintains the software. **Software engineering** is the technique that a software engineer chooses to produce an efficient, cost effective product.

Software life cycle includes requirement, specification, design, implementation, unit testing, integration, system testing, and maintenance which is called **water fall**.[3] For a complex software, a software engineer can use combination of water fall and recursive methodology to accomplish a task.

As a software engineer or a programmer you always ask yourself two important questions:

1. Software validation: Am I building the right product?
2. Software verification: Am I building the product correctly?
3. **Quality assurance** (**QA**) means that the product is both validated and verified for all test conditions.

1.4 Algorithm, Pseudo Code, and Flow Chart

> "An algorithm specifies a series of steps that performs a particular computation of a task."[4]

It is identifying the number of tasks and breaking down each task. Solve each task to make sure that each meets the requirements, and then connect all tasks to make sure the algorithm works for all cases.

> "**Pseudocode** is a mixture of English statements and C++ like control structures that can be translated easily into C++."[5]

Flowchart, like pseudo code, is used to represent an algorithm.

An algorithm is similar to writing a recipe for a salad dressing. I was watching TV and noticed that Dr. Abolfazly, a nutritionist, gave a salad dressing recipe in which he strongly recommended how healthy it is. I did not take any notes, but I tried to make it as follows the first time:

1/2 cup of honey

1 cup of apple cider vinegar

1 cup of olive oil

juice from 3 limes

4 oz ginger

4 oz of garlic

½ lb red onion

1 red bell pepper

2 spoons of dillweed

1 spoon of turmeric

1 spoon sea salt

2. Abraham Silberschatz, Peter Galvin, and Greg Gagne, *Operating Systems,* 8th ed. (New York: Wiley, 2009), 3.
3. Ian Summerville, *Software Engineering*, 4th ed. (New York: Addison Wesley, 2007), 12.
4. https://fiftyexamples.readthedocs.org/en/latest/
5. Nell Dale and Chip Weems, *Programming and Problem Solving with C++* (Burlington, MA: Jones and Bartlett Learning, LLC, An Ascend Learning Company, 2014), 166.

I put all of them in the blender and made the first version of the salad dressing. I tasted it but did not like the taste of ginger, onion, and garlic.

I made second version in which I added mint, basil, comino seed, flax seed, coconut oil, jalapeno, parsley, and cilantro. I added these because of their health benefits and also to reduce the smell of the onion, garlic, and ginger. This version was much better than the first.. I continued making the salad dressing, using all of the above ingredients, but increasing the amount of some of the ingredients and decreasing some others until the salad dressing was tasty for me.

In case of salad dressing, its health benefit and its taste are important for the user. However, in programming, validation, verification, and efficiency of the program is important. The program should also be well designed and documented, in addition to being user friendly.

An algorithm has to be written in enough detail that covers all the requirements and specifications. First you write the major tasks of the problem, and then you break each task into one or more tasks. Once your

algorithm is complete, you can implement it using any programming language. As a beginner, you design and trace it on the paper. If your design works for all cases, then you type it; otherwise you keep working on your design until it works for all cases. If your algorithm is done correctly based on the requirements, then implementation of it should be very simple.

How to start writing an algorithm?

1. Number of times each task should be done
   - 1.1 Either it performs once for all tasks,
   - 1.2 or it will be repeated more than once. This repetition is called loop. There are three kinds of loop:
     - 1.2.1 for loop
     - 1.2.2 while loop
     - 1.2.3 do…while loop
2. Detect and find the major tasks or functions.
3. Break down each task further to handle all cases for that particular task.
4. Highlight the key words such as the following:
   - 4.1 Read or get (These are inputs.)
   - 4.2 Compute or find (to do calculation)
   - 4.3 If …then…otherwise (making decision)
   - 4.4 Display, which produces output
   - 4.5 AND… OR …NOT (logical operator)
   - 4.6 Greater than, less than, equal, greater than or equal, less than or equal, and not equal (Relational operators)
   - 4.7 Multiply, divide, subtract, add, power, increment, decrement, absolute, sine,…(mathematical operations)
   - 4.8 To store one item (variable)
   - 4.9 To store more than one item which are related (array)
   - 4.10 To store all information about one person, employee, student (Record)
   - 4.11 To store more than one record (array of records)
   - 4.12 To access indirectly (pointer or reference)

Example 1: Write the steps of what you did since you woke up this morning and came to school. Break down each step even further, and explain in detail. For this example, the following five major tasks are considered:

1. Woke up
2. Took a shower
3. Ate breakfast
4. Used transportation to come to school
5. Went to the class

These five steps are abstract and general. Each of these steps can be broken down further and can vary from one person to another.

1.1 Alarm was set at 6:00 a.m.

1.2 Ignored alarm.

1.3 Alarm went off after five minutes.

1.4 Ignored again.

1.5 Woke up.

2.2.1 If male, the person then shaved and took a shower. If female, the person took a shower and then put on make-up.

3.1 Boiled eggs.

3.2 Toasted bread.

3.3 Made tea or coffee.

3.4 Ate breakfast.

4.1 Used own car, public transportation, bicycle, or walked to school.

4.1.1 If the person owned car, then he/she checked the following:

4.1.1.1 Gas

4.1.1.2 Oil

4.1.1.3 Water

4.1.1.4 Air

4.1.1.5 Drove to school

4.1.2 Or, if the person used public transportation, he/she did the following:

4.1.2.1 Paid

4.1.2.2 Came to school

4.1.3 Or if the person used a bicycle,

4.1.3.1 Checked air

4.1.3.2 Rode bicycle to school

4.1.4 Or, walked to school

5.1 Went to the right class

5.2 Made sure to be on time

5.3 Have a written excuse if absent or late

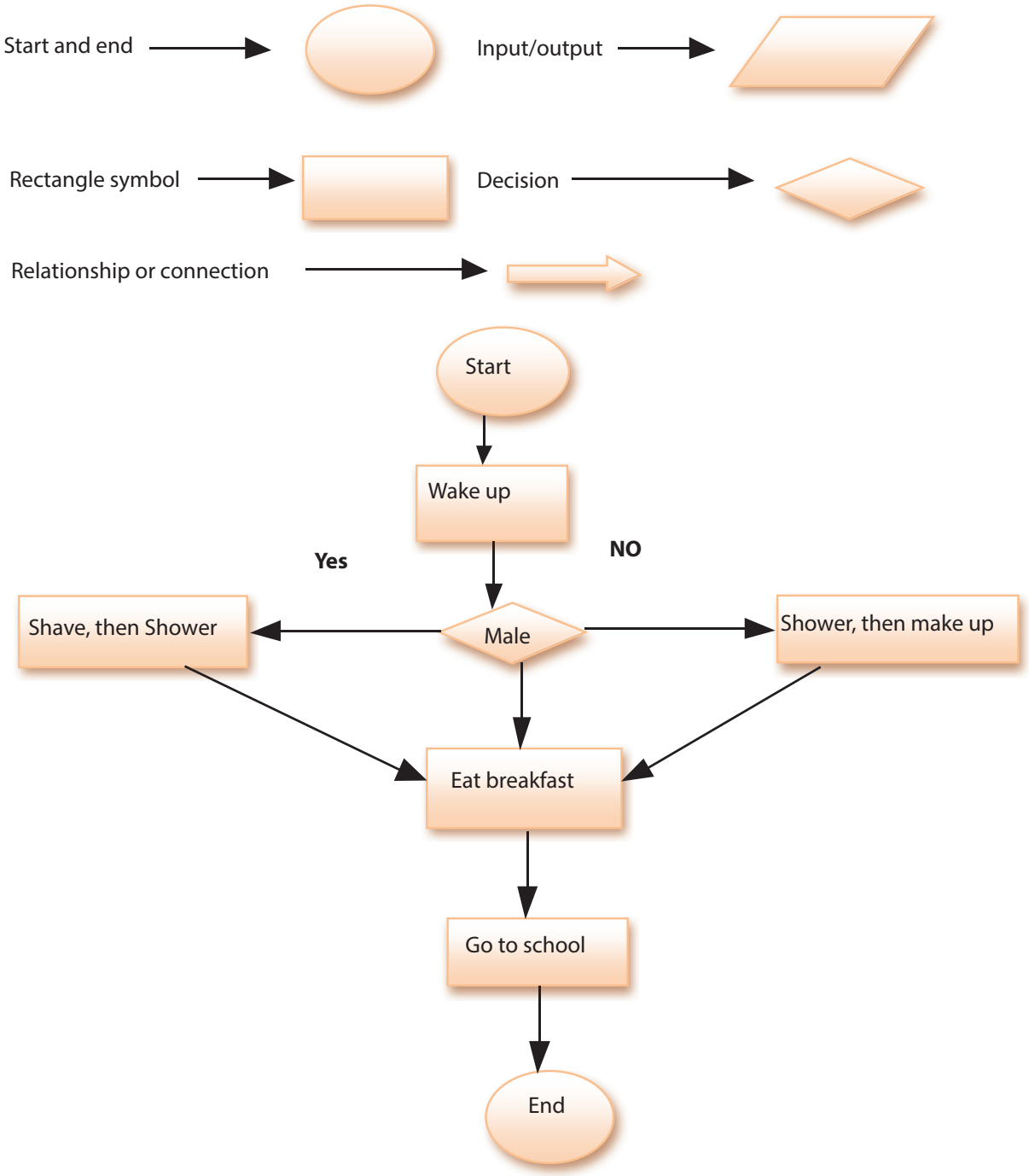


Figure 1.1 A Flowchart's Diagrams and Symbols[6]

6. "Flowchart Symbols and Their Usage," *Creately.com*. Copyright 2008–2015, accessed October 12, 2014. http://creately.com/diagram-type/objects/flowchart

Example 2: Write a program to read N employees weekly hours and rate per hour. Compute the gross salary. If gross salary is greater than $1000 ***and*** less than $2000, then deduct 28% tax; if gross salary is greater than or equal than $2000, then deduct 33% tax. Otherwise, deduct 22% tax. Computer tax, and compute net salary. Display weekly hours, rate per hour, gross salary, tax, and net salary.

Solution:

The number of repetition is highlighted in green.

The major tasks for this problem are underlined and highlighted in red.

The key word are highlighted in blue.

Algorithm:

Step 1: Identifying number of employees (green) and number of tasks (red)

Beginning Algorithm for Step 1

Loop: Repeat the tasks one at a time.

1. Get or read information.
2. Compute gross salary.
3. Compute tax.
4. Compute net salary.
5. Display.
6. Check counter that counts how employees' information has been read. If it is less than or equal to N, then go to the **loop**.

End of Algorithm for Step 1

Step 2: Breaking down each task and use the key words (BLUE)

Beginning of Algorithm for Step 2

Initialize counter to zero.

Loop:

1. Get or read information.
   - 1.1 Read or get weekly hours.
   - 1.2 Read or get rate per hour.
2. Compute gross salary.
   - 2.1 Multiply weekly hours by rate per hour to get gross salary.
3. Compute tax.
   - 3.1 If gross salary is greater than $1000 and less than $2000, then tax equals gross salary multiplied by 0.28.
   - 3.2 If gross salary is greater than or equal than $2000, then tax equals gross salary multiplied by 0.33.
   - 3.3 Otherwise, tax equals gross salary multiplied by 0.22.
4. Compute net salary.
   - 4.1 Net salary equals gross salary minus tax.
5. Display.
   - 5.1 Display weekly hours.
   - 5.2 Display rate per hour.
   - 5.3 Display gross salary.
   - 5.4 Display tax.
   - 5.5 Display net salary.

6. Counter
   6.1 Increase increment counter by one
   6.2 If counter is less than or equal to N, then go to loop.

End of algorithm for Step 2

This algorithm could be translated into any programming language. **No two designs should be the same.**

The following flowchart could represent the same sequence of operation as the above algorithm:

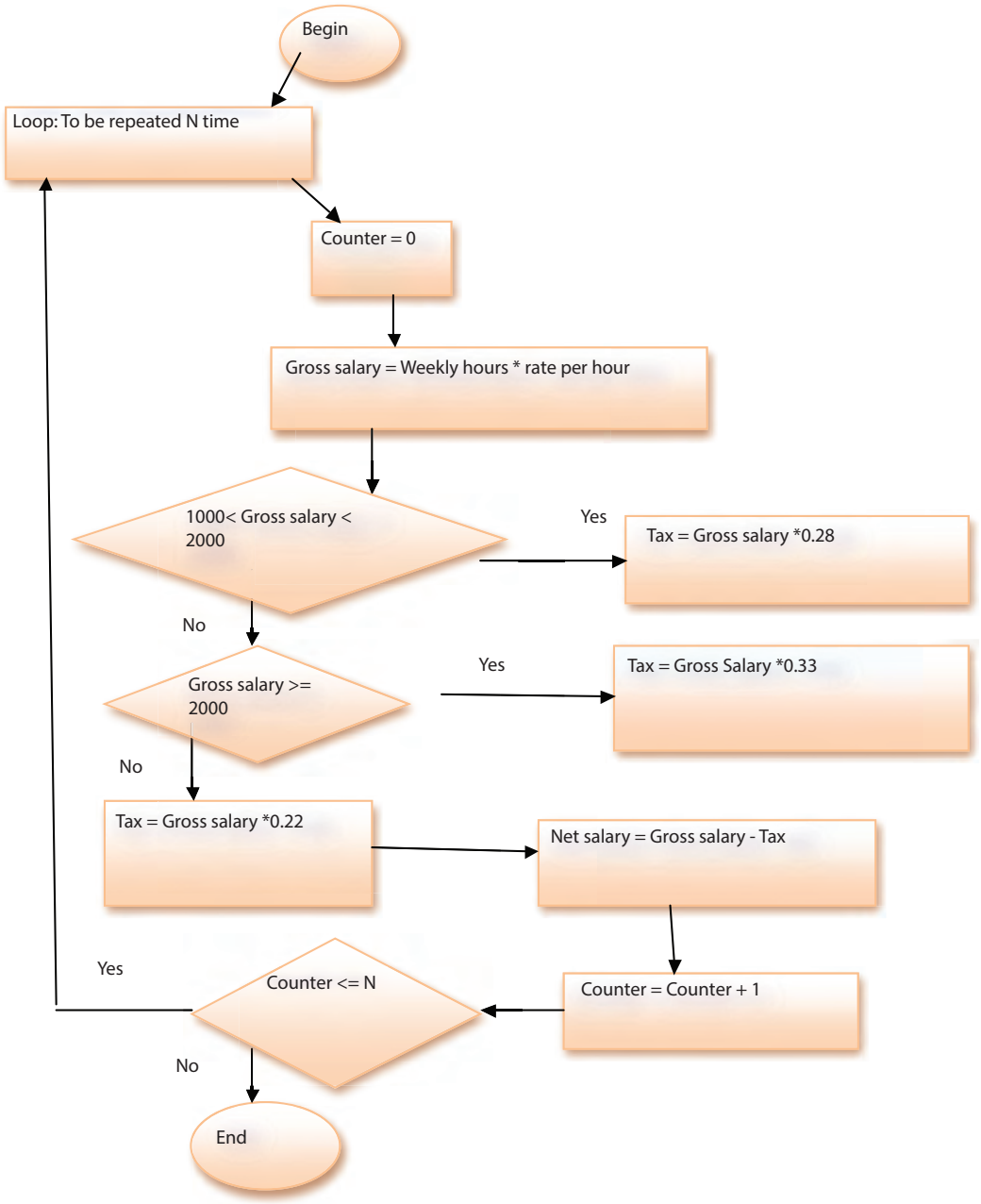


Figure 1.2

1.5 Programming Languages

Without programming languages, a computer is just a dummy machine. Programming language is needed to communicate with computer hardware.

Data is the information we give to or receive from a computer.

Record is a collection of data.

File is a collection of records.

Database is a collection of files.

Machine language is the low level language which is string of 0's and 1's.

High-level languages (HLL) are written more like English which is easy to read and understand. Examples of HLL are FORTRAN, PASCAL, C, and PL1.

The **C language** can also be considered as a middle level language because of its ability to access the low level.

Object-oriented programming (**OOP**) deals with abstract data types, information hiding, inheritance, overloading, and polymorphism.

Source file is the term for where programs written in HLL or OOL are stored.

Compiler: Programs written in HHL or OOP are more like English and cannot be understood by computer. Suppose I speak only Farsi and I want to communicate with someone who speaks only English. We cannot communicate with each other unless we have a translator who knows both Farsi and English. The same analogy can be used for programs written in any language other than "machine language." A compiler translates programs written in any language other than machine language to machine language.

Object file: A translated source file by the compiler is stored in a file called **object file**.

Executable file (**EXE**): Related object files can be linked together and create an executable file.

1.6 C++ Language

> "C was originally designed for and implemented on the UNIX operating system on the DEC PDP-11, by Dennis Ritchie."[7]

The name of C language was taken from the second letter of BCPL.

C++ language was designed and implemented at AT&T Bell Laboratories by Bjarne Stroustrup in 1983. C++ is a tool for object-oriented programming which supports data abstraction, inheritance, overloading, exceptions, and polymorphism. The object is a variable that has a type and storage class which can hold information of its type. C++ is superset of C. C++ supports features of OOP.

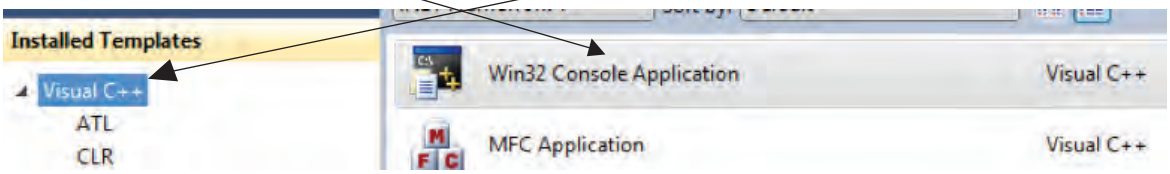
> "The name C++ (C plus plus) was coined by Rick Macitti in the summer of 1983. The name signifies the evolutionary nature of the changes from C. " + + is the C increment operator."[8]

7. Brian W. Kernighan and Dennis M. Ritchie *The C Programming Language*, Second Edition (New York: Prentice Hall, 1988), xi.
8. Bjarne Stroustrup, *The C++ Programming Language*, Second Edition (Reading MA: Addison Wesley), 4.

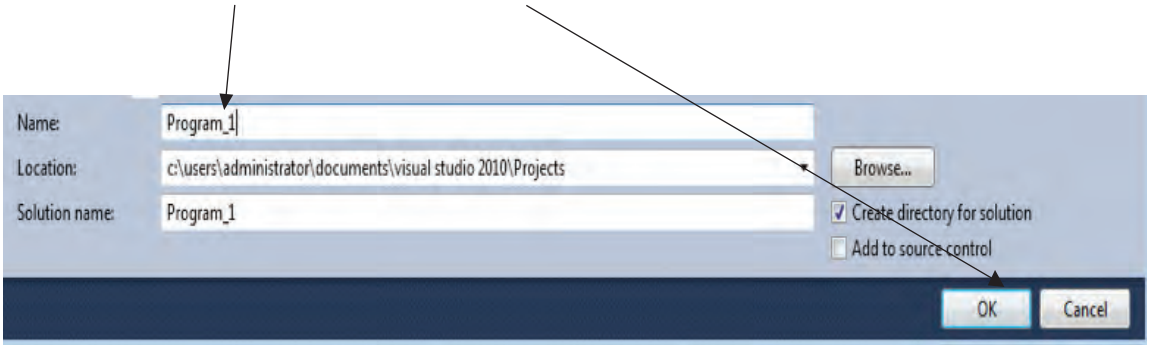
If you are using Microsoft Visual Studio, then following are the steps to start your program:

1. Click on Microsoft Visual Studio icon.

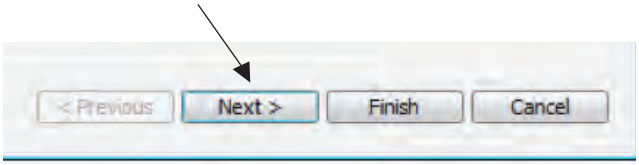
2. Select "New Project."
3. Select Win 32 Console Application (make sure Visual C++is highlighted).

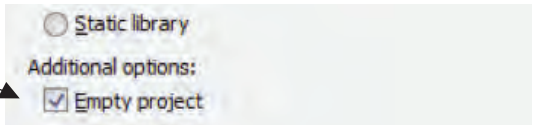


4. Enter project name and then click "OK."



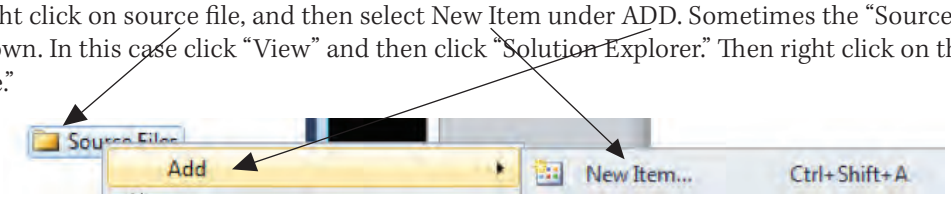
5. Click "Next" and then check "Empty project."



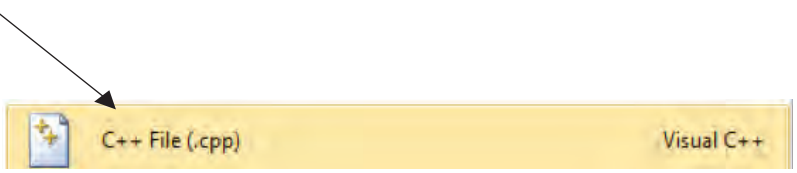


6. Click "Finish."

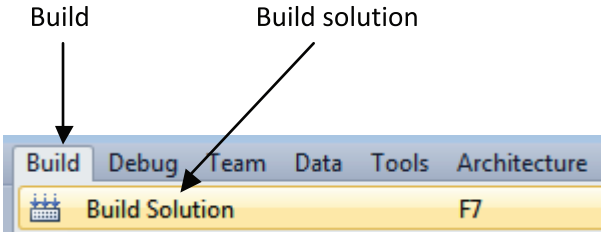
7. Right click on source file, and then select New Item under ADD. Sometimes the "Source File" is not shown. In this case click "View" and then click "Solution Explorer." Then right click on the "Source File."



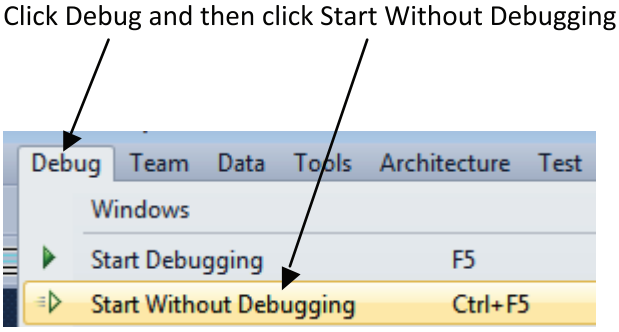
8. Select "C++ File (.cpp)"



9. Enter Project Name and then click "Add."
10. At this time you can start typing your program.
11. Once you have finished typing, you need to compile your program. To do that, click on "Build" and then click on "Build Solution." At this time, if your program is error free, then it is ready for execution and testing.



12. Click "debug" and then click "Start Without Debugging." From here, you follow the instruction of your own program or design, which means interacting with your program.



1.7 Documentation

Documentation should have the following format:

a. The purpose of the program or function: Explain in detail what the requirement of the program or function is.

b. Inputs: Explain what the inputs are.

c. Outputs: Explain what the outputs of program are expected to be according to the requirements.

d. Algorithm of the program, either written in pseudo code or using a flowchart, should be transformed or translated into any programming language.

e. Test condition(s): The program should be tested for all cases.

f. Exception(s): Explain what the exception(s) is/are and the exception(s) should be handled.

Note that each line of code must be documented and explained.

1.8 Programming in C++

Program 1a

Write a program to display your name.

```
// Purpose: To display my name
// Input: None
//Output: My name
//Algorithm: Use cout to display my name.
//Exception: None

#include <iostream>

using namespace std;
int main(void)
{
        // Display My name
        cout<<"\n\t My name is Danial \n"<<endl;
        system ("pause");
        return 0;
}
```

```
My name is Danial
```

Output of the program

Detailed explanation of the above program

1. ***Comment*** is used to explain what the problem is and to make program more readable. The symbol for comment is " // "
2. ***#include*** directive or preprocessor directive: Allows any files followed by this directive to be known by the compiler. Two types of file can be used with include directive.
   a. **#include <system defined file>** which allows system defined file to be known to the compiler
   b. **#include "user defined file"** which allows user defined file to be known to the compiler.
3. ***Std*** allows all objects in ***iostraem***, including ***cout***, ***cin***, and ***endl***, to be accessed directly.
4. int main(void)
   4.1 ***Int*** stands for **integer** which is a data type that represents all whole numbers—including positive, zero, and negative.
   4.2 Placing **int** before the main function informs the compiler that the main function would return a value which is type of integer.
   4.3 The name of main function in C++ is ***main***.
   4.4 Main(***void***) means that this function does not accept any input or parameter. If a function accepts input, then both input's name and the input's type should be specified. If there are more than one input, then each input is separated by a comma ",".
5. Open curly bracket, {, means beginning of the main function or beginning of a block of statements.
6. Indentation is used to make program clearer and more readable.
7. Part of iostream which will display the exact text typed between two double quotation mark (**"..."**) is ***cout***.
8. ***Endl*** **(end of line)** means advance the pointer to the next line so the next text would be displayed on a new line. Endl is analogous to "pressing the enter key on the keyboard".
9. A **semicolon ;** at the end of each statement means end of that statement.
10. ***n*** would do the same job as **endl**.
11. **t** is analogous to pressing the Tab key.
12. ***Return 0*** means that this function, or program, return a value zero which is type of integer. Because no operation is done in this program, it just displayed the day of a week.
13. ***System ("pause")*** allows user to see output by pausing till user hit any key.
14. Close curly bracket **"}"** mean end of the main function or end of a block of statements.

Programming is both an art and a design. You should really enjoy designing your program. Programming requires time and patience. If you are unwilling to spend time and you are not patient, then you will have a problem really learning the concept in depth. The more you practice, the easier it becomes—and the more you will enjoy programming. Programming, like art, is your own creation. Both your program and the output should reflect your artistic and design ability.

Program 1b

Write a C++ program to display the days of a week using namespace std.

```
//  Purpose: This program displays the day of a week.
//Inputs: None because this program does not get input, and this is the first programming in C++
//Outputs: Display seven days of the week.
//Algorithm: Print each day of the week on a separate line.
//Test condition(S): None
//Exception: None
#include <iostream>
using namespace std;
int main(void)
{
        // Display Monday
        cout<<" Monday"<<endl;
        // Display Tuesday
        cout<<" Tuesday \n";
        // Display Wednesday
        cout<<" Wednesday"<<endl;
        // Display Thursday
        cout<<" Thursday"<<endl;
        // Display Friday
        cout<<" Friday"<<endl;
        // Display Saturday
        cout<<" Saturday"<<endl;
        // Display Sunday
        cout<<" Sunday";
        cout   <<endl;
         system ("pause");
         return 0;
}
```

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Press any key to continue . . .
```

Output of the program

Program 1c

Write a C++ program to display the days of a week using std object.

```
// Purpose: This program displays the days of the week by excluding using namespace std
        // statement and using std object and scope resolution "::"
//Inputs: None because this program does not get input, and this is the first programming in C++
//Outputs: Display seven days of the week.
//Algorithm: Print each day of the week on a separate line.
//Test condition(s): None
//Exception: None
```

```
#include<iostream>
//using namespace std;
int main(void)
{
        // using std and scope resolution :: to access cout object
        std::cout<<" Monday\n" ;
        std::cout<<" Tuesday \n" ;
        std::cout<<" Wednesday \n" ;
        std::cout<<" Thursday \n";
        std::cout<<" Friday \n";
        std::cout<<" Saturday "<<std::endl;
        std::cout<<" Sunday \n";
        return 0;
}
```

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Press any key to continue . . .
```

Output of the program

The output of both programs is the same; however, in program 2 using namespace std is commented out, meaning that compiler ignores it. Therefore cout and endl object cannot be used directly. Std is using scope resolution symbol "::" (std::cout and std::endl) to access ***cout*** and ***endl*** objects.

Program 1d

This program displays the days of a week and uses system ("cls") and system ("pause").

```
// system ("pause") Function waits for user to press the enter key to continue.
// the execution of the program
// system ('cls") Function would clear the screen.

#include<iostream>
using namespace std;
int main(void)
{
        cout<<" Monday"<<endl;
        cout<<" Tuesday \n";
        cout<<" Wednesday"<<endl;
        cout<<" Thursday"<<endl;
        system("pause");
        system ("cls");
        system("pause");
        cout<<" Friday"<<endl;
        cout<<" Saturday"<<endl;
        cout<<" Sunday"<<endl;
        return 0;
}
```

Because of system ("pause") statement, user has to press any key to continue the execution.

```
 Monday
 Tuesday
 Wednesday
 Thursday
Press any key to continue . . .
```

Output of the program

After pressing any key, the system ("cls") statement will clear the screen.

The program will produce the remaining output, which would be Friday through Sunday.

```
Press any key to continue . . .
 Friday
 Saturday
 Sunday
Press any key to continue . . .
```

Summary

A computer is a dummy machine that has to be programmed to perform tasks.

Main components of computer hardware consists of input devices, output devices, memory, and central processing unit.

Program: A list of instructions

Bit: Information stored in memory are strings of 0's and 1's. Each 0 or 1 is called a **bit**.

Byte: A collection of eight bits is **one byte**, which represents one character.

Computer software is a collection of programs.

Software is a list of programs.

The **operating system**, which is a software program, makes a dummy machine smart.

Software engineering is a technique that a software engineer chooses to produce an efficient and cost effective product.

Software life cycles are requirement, specification, design, implementation, unit testing, integration, system testing, and maintenance.

Algorithm is step by step defining and solving the problem.

Programming languages are the languages, either in machine language or translated into machine language, which are needed to communicate with computer hardware to perform real world application.

Data is the information we give to or receive from a computer.

Record is a collection of data.

File is a collection of records.

Database is a collection of files.

Machine language is the low level language that is string of 0's and 1's.

High level languages (HLL) are more like English. Examples of HLL include FORTRAN, PASCAL, C, and PL1.

The **C language** can also be considered as middle level languages because of its ability to access the low level language.

Object-oriented programming (**OOP**) deals with abstract data types, inheritance, information hiding, and polymorphism.

Source file: Programs written in HLL or OOL are stored in a file called **source file**.

Compiler translates program written in any language other than machine language to machine language.

Object file: Translated source file by compiler is stored in a file called **object file**.

Executable file (**EXE**): Related object files can be linked together and executed to create an executable file.

Comment is used to explain what the problem is and to make program more readable. The symbol for comment is " // "

#include allows any files followed by this directive to be known by the compiler.

Std allows all objects in iostream—including cout, cin, and endl—to be accessed directly.

int stands for integer which is a data type representing all whole numbers including positive, zero, and negative. If placed before main function, ***int*** tells the compiler that the main function return a value which is type of integer.

The name of main function in C++ is ***main***.

The term main (***void***) means that this function does not accept any input or parameter.

Open curly bracket **"{"**means begin of the main function.

Indentation is used to make the program clearer and more readable.

The part of iostream which will display the exact text typed between two double quotation mark ("…") is ***cout***.

The ***endl*** (**end of line**) means advance the pointer to the next line so the next text would be displayed on a new line.

Semicolon (**;**) at the end of each statement means end of that statement.

n would do the same job as **endl**.

t would do the same job as the Tab key.

Return 0 means that this function or program returns a value 0. Because no operation is done in this program, it just displayed the day of a week.

System ("pause") allows user to see output by pausing till user hit any key.

Close curly bracket **"}"** mean end of the main function.

Exercises

1. What are the components of computer hardware?
2. What is a program?
3. What is software?
4. Define software engineer?
5. What is software engineering?
6. Give an example of an algorithm.
7. What are symbols for a flow chart? Give an example of flowchart.
8. Define memory, CPU, ALU, CU, input devices, and output devices.
9. Define machine language, high-level language, and object-oriented language.
10. Write the history of C++ and its features.
11. Define compiler, source file, object file, and executable file.
12. Explain documentation policy.
13. How do you display information in C++?
14. Explain the purpose of **" return 0'** in the program.
15. What does **" int main (void)** " mean?
16. What is the symbol for comment?
17. What is the symbol for carriage return?
18. What is the symbol for the Tab key?
19. Explain " **#include "** directive.
20. What is the advantage of using "using namespace std;" in the program?
21. What does **" #include <iostream>"** do?
22. Explain how **" cout "** works. Give an example.
23. What do **system ("pause") and system ("cls")** do?
24. What are the purposes of open curly bracket ({) and closed curly bracket (}) ?
25. Write an algorithm to find the average grade of each course for a student who is taking four courses. There are five exams per course.
26. Write an algorithm to find the letter grade and GPA for three students, each of whom takes four courses. There are five exams per course that worth 6 percent of the grade, plus eight homework assignments worth 40 percent of the grade. The letter grade for each course is as follows:

    An A equals 90–100, 80–89 is a B, 60–79 is a C; 60–69 is a D, and 0–59 equals F.

    The value of each letter grade is as follows for GPA calculation:

    A = 4, B = 3, C = 2, D = 1, and F = 0.

Programming Practices

1. Write a C++ program to display the following shape:

```
**
******
**********
******
**
```

2. Write a C++ program to display your name.
3. Write a C++ program to display your school name.
4. Write a C++ program, on three different lines, to display your name, your major, and school name.
5. Type, compile, and run the following program. Explain the output based on your observation.

```
#include<iostream>
using namespace std;
int main(void)
  {
        cout<<" Monday"<<endl;
        cout<<" Tuesday \n";
        system("pause");
        system ("cls");

        cout<<" Wednesday"<<endl;
        cout<<" Thursday"<<endl;
        system("pause");
        system ("cls");

        system("pause");
        cout<<" Friday"<<endl;
        cout<<" Saturday"<<endl;
        cout<<" Sunday"<<endl;

        system("pause");
        return 0;
}
```

6. Use the following program to do the following:

   Find how many errors there are. What are those errors? Correct the program; then type, compile, and run the corrected program.

```
#include “iostream”
using name space std
main( )
    {
        Cout << January";
        Cout <<"\t February   \t  March\n";
        system("pause")
        cout<<"\n\t April"<<endl;
        cout<<" \t May    \t June ";
        system  "pause";
        cout<<" \n\t July"<<endl;
        cout<<" \t August"<<endl;
        cout<<" \t September \n";
        System ("pause");
cout<<" \t October    \t                November"<<endl;
        cout<<" December \n";
        system("pause");
        Return 0;
    }
```

Employee

7. Write a C++ program to display an employee's weekly hours = 40.00 and rate = 25.00.

Project

8. Write a C++ program to use ***cout*** to display a triangle.