Study Guide

For

Fundamentals of Programming

Department of Computer Science

Faculty of Informatics

Prepared by

Sebsibe Hailemariam (PhD)

January 2012

Summary

Fundamental of programming is a course offered for junior students at the Department of Computer Science and other computing field of study. The course can be seen as a corner stone to Computer Science students as it provides diverse insight into the field of the Science. In the first place, this course indicates what students are going to do in the future after completion of their course. Secondly, this course gives broader view on how computer solve users problem which is also one of the intended profile of students to attain upon completion of the program. Moreover, the course allows students to see briefly how computer function in order to do some task.

This guideline is not a reference that students should use during the period where the course fundamental of programming is offered. Rather it can be used for the students as a checklist to know the expectations that students are expected to know and what they already know. Hence, this guideline clearly shows for the students' existence of any gap among the expected knowledge, skill and attitude to be achieved at the end of the course.

In addition, this guide line is relevant not only to students but also for instructors that teaches this course. Instructors should use this guide line as a check list to see their periodic status while teaching the course and plan a head how to cover the important topics identified that instructors are expected to teach students and transfer knowledge and skill to their students.

This guideline is an important starting point to someone who wants to learn programming with independent study. Finally, the guide line is an important asset to the Department to give a direction to instructors what needs to be covered while he/she is teaching the course.

Unit one

Computer Systems and Programming Language

Summary

Programming is an art of instructing what the computer hardware should do in order to fulfill the needs and requirements of users. Programmers can write efficient and high quality programs if they know important elements of the computer hardware and the software that run under the computer system for various reasons. This chapter is intended to give a direction to students what they should know about the computer hardware and software to be successful programmer.

General Objective

The purpose of this unit is to revise basic computer system components and the role of the components in solving problem by programming and to develop algorithmic thinking so that students can translate a given problem into algorithm that can be implemented using computer programming language.

Specific Objectives

At the end of this unit, students should be able to:

- Describe the different hardware components of the computer system and their functionalities
- Distinguish Input unit, output units, RAM, secondary storage devices, CPU, Arithmetic Logical Units, Control Unit, Register, Cache Memory, communication bus, (Address Bus, Data Bus, and other communication systems that links different components of the computer hardware)
- Identify the role of these hardware components
- Describe computer software
- Identify the classification scheme of software
- Distinguish computer software from computer hardware
- Explain what an Operating System is
- Identify the role of a computer operating system
- Distinguish the different types of operating systems
- Describe what language software is
- Distinguish the different types of language software

- Identify the different generations of language software
- Explain what is application software is
- Describe how to solve problem using Computer
- Describe the steps to solve problem using computer
- Distinguish the different phases in solving problem using computer (Understanding the problem, designing the algorithm, implement the algorithm, debugging and testing the program, deploying the solution)
- Describe how to develop algorithm using flow chart or pseudo-code
- Describe the translation procedure of source code into machine understandable code
- Describe how to translate source code into machine code for different types of source codes (Assembly language code, compiled language code or interpreted language codes)
- Distinguish the difference among the different translators (Assembler, Interpreter and Compiler)
- Identify some sample programming languages which are compiled or interpreted types
- Describe the steps from writing a code in compiled language until execution
- Describe the concepts of pre-processing, compiling, linking, loading, executing
- Describe the role of the operating system while the loader loads an executable program
- Distinguish code segment and data segment in the RAM for a process?
- Describe the three types of programming error?
- Identify the basic characteristic of syntax (compiled type) error?
- Identify the basic characteristic of logical error?
- Identify the basic characteristic of the runtime error?

Self-test Questions without answer key

Attempt all of these questions and make sure that you answer them correctly.

1. Explain the role of the basic components of a computer hardware: RAM , Register, Cache memory, Address bus, Data bus, ALU, Control unit, Secondary storage device, etc
2. What is clock cycle of a CPU? How many clock cycle an instruction will take?

3. Compare & contrast speed of fetching data by the CPU form different data source location such as Register, Cache memory (L_1,L_2 & L_3), RAM, Hard disk, Flash disk, CD, Magnetic tape.
4. Explain clearly about pre-processing, Compiling, Linking, Loading and Executing.
5. Explain why OS assign code segment, Data Segment, Heap Segment & Stack Segment to each program loaded to be executed by the CPU.
6. Write a flowchart that accept a number and check whether the number is even or odd
7. Write a flowchart that search a given searchKey value is in the list of values
8. Write a flow chart that checks whether a given number is prime or not. A prime number is a number that has only two distinct factors (1 and the number itself)
9. Write a flowchart that search the smallest prime number greater than or equals to N.
10. Write a flowchart that searches the N^{th} prime number.
11. Discuss the different types of pre-processing directives.
12. It has been decided that a bonus of 12% is to be given for each employee in an organization. It was also agreed that if an employee has worked for more than 13 years, s/he is to receive an additional amount of Birr 350.00. Draw a flaw chart that describes how the bonus for an employee is calculated given his/her current salary.
13. Write a flowchart that display sequence of n positive integers starting from a user specified number m
14. Write a flowchart that convert a mark for a course to its corresponding letter-grade using the following scale

Mark	Grade	Mark	Grade
>=90	A+	>=55	C+
>=80	A	>=45	C
>=75	B+	>=30	D
>=60	B	<30	F

15. Write a flowchart that compute and display the sum of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.
16. Write a flowchart that compute and display the product of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.

17. Write a flowchart that accept two numbers and an operator and compute and display the result of the expression (be careful to handle division by zero)
18. Write a flowchart that accept N integers and display them in reverse order
19. Write flowchart that computes the number of days in a given month and year of a Gregorian calendar.
20. Write a flowchart that compute and display the next date of a specific date in European calendar accepted from the user. You must consider leap year condition along with other conditions.
21. Write a flowchart that accept a list of integers different from zero and terminate accepting the integers when the input is zero.
22. Modify the above flowchart so that it would display the number of positive and negative integers
23. Write a flowchart that calculate income tax of an employee given the total monthly salary from the keyboard and following the following personal tax rule of Ethiopia
    a. All salary amounts less than or equal to 150 Birr are tax free.
    b. Salary amounts between 151 to 650 birr will be deducted 10% of the salary – 15 birr
    c. Salary amounts between 651 to 1400 birr will be deducted 15% of the salary – 47.50 birr
    d. Salary amounts between 1401 to 2350 birr will be deducted 20% of the salary – 117.50 birr
    e. Salary amounts between 2351 to 3550 birr will be deducted 25% of the salary – 235 birr
    f. Salary amounts between 3551 to 5000 birr will be deducted 30% of the salary – 412.50 birr
    g. Salary amounts above 5000 birr will be deducted 35% of the salary – 662 birr
24. Write a flowchart that search and display the maximum and the minimum of a list of numbers entered by the user. Where the size of the list is to be entered by the user.

25. Write a flowchart that calculate and display the factorial of a number n where n is a positive number to be entered by the user.

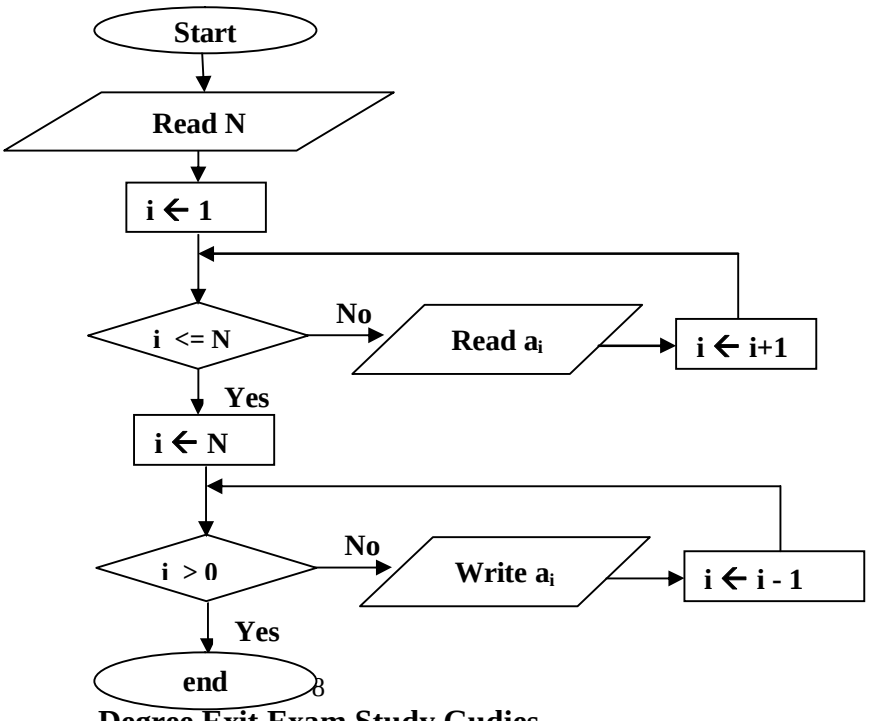
Questions with answer key

1. A code (Source program) written in C^{++} cannot be executed directly by the machine (Computer). Why? What needs to be done?
2. Give some list of C++ compilers
3. Explain compile time/syntax error, Logical & Run time errors
4. Design a flow chart to accept N numbers and display them in reverse order
5. Write a flowchart that accept a list of integers different from zero and terminate when the input is zero.
6. Write a flowchart that calculate income tax of an employee given the total monthly salary from the keyboard following the following personal tax rule of Ethiopia and display the tax on the screen
    a. All salary amounts less than or equal to 150 Birr are tax free.
    b. Salary amounts between 151 to 650 birr will be deducted 10% of the salary – 15 birr
    c. Salary amounts between 651 to 1400 birr will be deducted 15% of the salary – 47.50 birr
    d. Salary amounts between 1401 to 2350 birr will be deducted 20% of the salary – 117.50 birr
    e. Salary amounts between 2351 to 3550 birr will be deducted 25% of the salary – 235 birr
    f. Salary amounts between 3551 to 5000 birr will be deducted 30% of the salary – 412.50 birr
    g. Salary amounts above 5000 birr will be deducted 35% of the salary – 662 birr

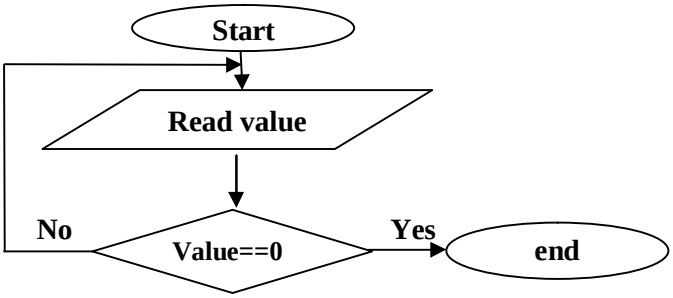
Answer key

1. *A code (Source program) written in C++ cannot be executed directly by the machine as the C++ code is not directly understandable by the machine. Machine code is codes written in zeros and ones where as C++ codes are written using sequences of ASCII characters. The C++ code needs to undergo a number of stages to be executed by the machine. First, the preprocessor must handle the source code to handle all preprocessing directives. The output of the preprocessor will be given to the compiler so that all instructions will be translated into the equivalent object code. The object code should be linked with other required object codes (can be pre-compiled library objected codes or user defined library object codes). The output the linker can be executed if the program is loaded into the RAM as the RAM is the working memory of the computer.*
2. *The following are some of the C++ compilers available for use in writing C++ program*

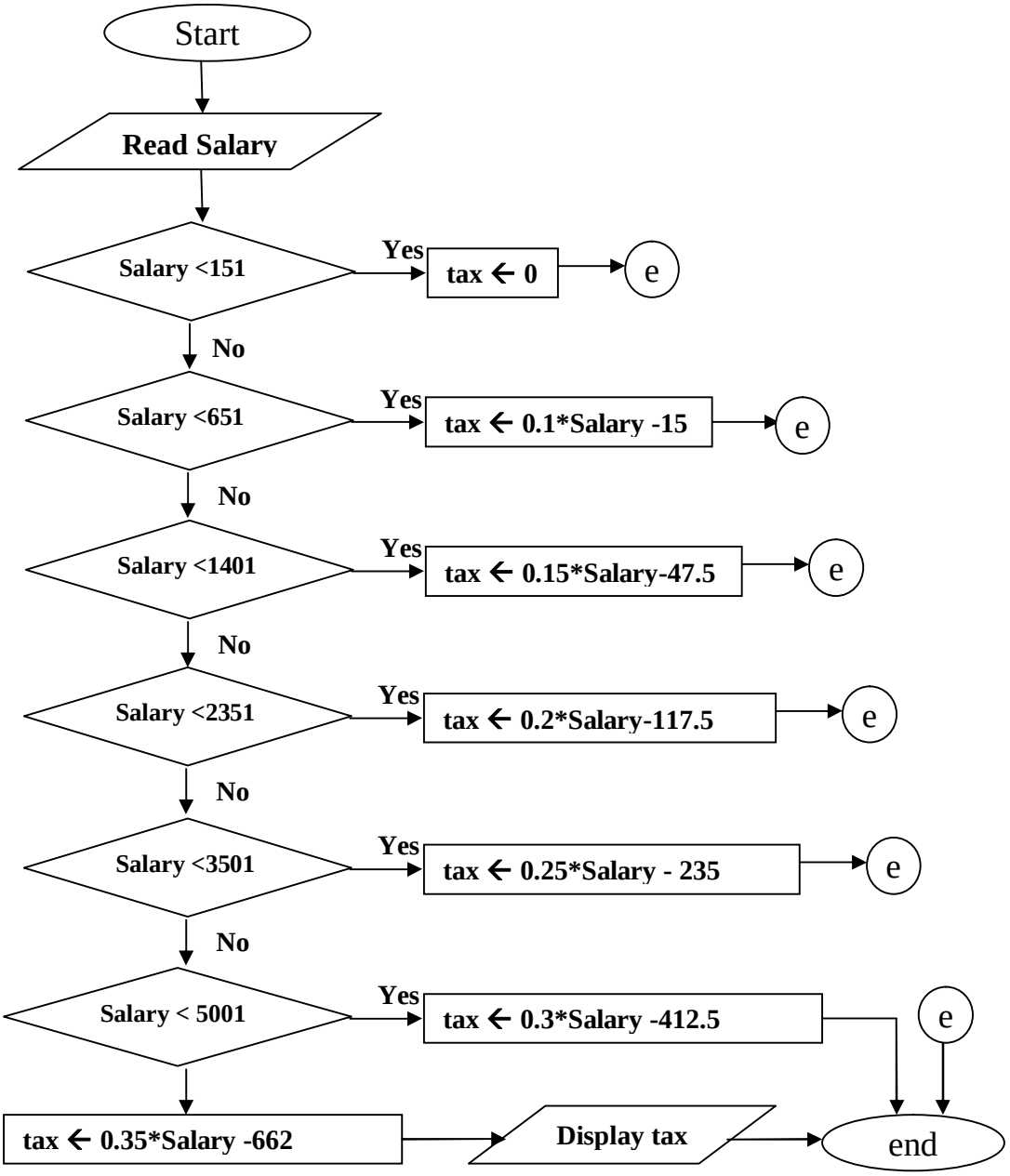
Borland C++	*GCC*
Turbo C++	*C++ Compiler professional*
Dev-C++	*Visual C++*

3. Syntax/compile time error is an error caused as a result of wrong lexical or syntactic elements of the program construct which cannot be translated by the compiler. Logical error is an error due to the computational logic of the program that results wrong output which may mislead users decision. Runtime error is an error that result abnormal termination of program execution.
4. 

5.


Start
Read value
No
Value==0
Yes
end

6.


Start
Read Salary
Salary <151
Yes
tax ← 0
e
No
Salary <651
Yes
tax ← 0.1*Salary -15
e
No
Salary <1401
Yes
tax ← 0.15*Salary-47.5
e
No
Salary <2351
Yes
tax ← 0.2*Salary-117.5
e
No
Salary <3501
Yes
tax ← 0.25*Salary - 235
e
No
Salary < 5001
Yes
tax ← 0.3*Salary -412.5
e
tax ← 0.35*Salary -662
Display tax
end

Unit Two

Basics of Programming Language

Summary

Program is a language written using selected programming language. Programming language as a language has its own lexical elements and statements constructed from the lexical elements. The statements of a program written in a selected programming language follow its own grammar. This is exactly the same as the natural language that we use in day to day communication such as Amharic, English, Afaan Oromo, Tigregna. These languages has words, numbers, punctuation marks, acronyms as lexical elements and grammar that define legal sequence of such lexical elements to form valid statement. Lexical terms has different role such as subject maker, object maker, linking clauses, etc and the same is true in programming language. A program is hence a sequence of statements that potentially instruct the computer hardware what to do to solve a problem.

General Objective

The purpose of this unit is to introduce student with general structure of a program in a programming language; simple grammar of basic statements and expressions; lexical elements and issues associated with them for use in statements/expressions of the selected programming language (C++ is chosen for this guideline).

Specific Objectives

At the end of this unit, students should be able to:

- Describe the structure of the C++ program
- Describe preprocessing directives
- Identify the most common preprocessing directives
- Use preprocessing directives
- Describe global declaration statements (Variable identifier declaration, Constant identifier declaration, Function declaration, External function declaration)
- Explain how many functions a C++ program can have
- Identify the elements of a function definition and implementation section
- Describe function return type, function name, function argument

- Distinguish local versus global variables
- Identify constant identifier
- Explain constant identifier declaration statement
- Identify what an identifiers is (one of the lexical element)
- Define an identifiers
- Identify keywords in C++ programming
- List examples of keywords in C++ and their purpose
- List some examples of non-keyword identifiers
- Discuss the purpose of identifier (As a function name, As a variable identifier, As a constant identifier, As a keyword, As user defined data type or class name)
- Describe declaration statement
- Identify the syntax of declaration statement
- Identify data type in declaration statement
- Identify the purpose of data type to compilers
- Show how to declare variable identifier with/without initialization
- Show how to declare constant identifier
- Describe expressions in C++ program
- Define an expression
- Identify the ingredients of an expression
- Identify the valid operands in an expression (Function calls, Variable identifiers, Constant identifiers, Literal constants)
- Identify the valid operators in an expression and their semantics (Arithmetic operators, Relational operators, Logical operators, Increment operator, Decrement operator, Bitwise operator, Conditional operator (?:), Comma operator, Sign operator (+ and -), Assignment operator, Parenthesis)
- Identify the unary, binary and tertiary operators in C++ programming language
- Describe operator precedence and show the operator precedence rule in C++
- Describe association rule of operators and state the association rule of the different operators
- Identify and use especial characters represented using escape sequence

- Describe assignment statements
- Identify the syntax for assignment statement
- Identify how assignment statement is evaluated
- Describe output statements
- Define an output statement
- Identify the syntax for output statement
- Show how to output string, expression value, etc
- Explain how output statement is evaluated
- Show some example of simple output statements
- Describe input statement
- Define input statement
- Identify the syntax for the input?
- Show sample example of simple input statement
- Describe the syntax for the main function definition
- Explain why we need data type before the function name (Is it optional or not)
- Identify the possible data types that can be used for the main function
- Explain whether the main function name is case sensitive or not
- Defend that the main function is compulsory for every C++ programs
- Explain the purpose of the parenthesis "( )" for the main function
- Describe what can be placed inside the parenthesis?
- Describe return statement inside a main function
- Identify why we need return statement in the main function
- Construct simple program consists of declaration statement, assignment statement, input and output statements)

Self-test Questions

1. Where the preprocessor finds the files needs to be included and defined using the #include preprocessor directive
2. Give at least four invalid identifier names each having different reasons for being invalid
3. Write three possible ways of writing a C++ program main function definition
4. Give a reason why the following sequence of characters are not identifiers

a. void
b. 1_test
c. Phone number
d. AmountIn%
e. _Salary@eth
f. asm

5. Given a declaration statement, **"double x = 7.0; char ch; float f; short int age; ",** how much memory space would be reserved for **X, ch, f, and age**
6. Show the order of evaluation for the following expressions
   a. Z = a && b || !c (assume a = 10; b = 0; c = -2)
   b. 4 + 5 * 2 / 4 * 0.5
7. Why the variable to the left of an assignment statement is a variable identifier?
8. Write a simple output statement that output the message **"Hello world"** on the screen
9. Write a simple output statement that output the message the value of the expression **(4+5*2 /4*0.5)** and the expression **(10 + Y * (Z - 5))** separated by **tab** character
10. Write a simple input statement used to read value for the three variable identifier X, Y and Z
11. A programmer wrote an input statement **cin>>N;** where N is a declared constant identifier. The compiler refused to compile this line of statement. Why do you think is that?
12. Write a program that display the message "Hello World!!" on the screen
13. Write a program that display the number from 1 to 4 separated by tab character
14. Write a program that display the number from 1 to 4 in a separate line using only one output statement
15. Write a program that reads one integer variable, one character and one double and display them in reverse order

Questions with answer key

1. Write a ***"preprocessor directive statement"*** that define MAX to be 1000
2. Write a preprocessor directive that includes **string.h**
3. Write a declaration statement that declares a float variable identifier called mass with initial value of zero.
4. Write a declaration statement that declares a constant identifier called gravity with value 9.8

5. Show the order of evaluation for the expression X = 10.5 + Y * (Z - 5) assuming the following value (X = 5; Y = 10; Z = 2.5). Note that the data type can be guessed from the current value given
6. Write a program that accept three floating point numbers and display their sum

Answer key

1. ***#define MAX 100***
2. ***#include<string.h>***
3. ***float mass=0;***
4. ***const float gravity 9.8;***
5. X = 10 + Y * (Z - 5)
    a. First evaluate the subtraction operator that results X = 10.5 + Y * 2.5
    b. Next evaluate the multiplication operator that results X = 10.5 + 25.0
    c. Next evaluate the addition operator that results X = 35.5
    d. Finally the value of the expression which is 35.5 will be assigned to X. However, as X is an integer, 35.5 will be first truncated into 35 and 35 is assigned to X.

6.

```
#include<iostream.h>
int main()
{
        float f1,f2,f3;
        cout<<"Enter the three float numbers  ";
        cin>>f1>>f2>>f3;
        cout<<"The sum of the three float number  =  "<<f1+f2+f3<<endl;
        return 0;
```

Unit Three

Control Statements

Summary

Solving problem by computer needs computation of finite set of instructions step by step in a well-controlled fashion. Usually instructions are placed in a program ordered by line number and from left to right if more than one instruction is placed in one line. However, the execution plan to solve a given problem may require executing the current instruction to determine the next instruction to be executed. Hence, the next instruction to be executed cannot be determined just by looking the current instruction line number. Instructions that determine what to be the next instruction to be executed after they get executed are called control instructions/statements. If the current instruction is not a control statement, then the next instruction that would be executed is the one which is located next to the current statement in their logical ordering (line number and left to right in a line). This unit focuses on providing students a guide about control flow and control statements as well as their computations strength.

General Objective

The purpose of this unit is to introduce student with control statements and structures appropriate for the different possible control statements.

Specific Objectives

At the end of this unit, students should be able to:

- Describe control flow in programming
- Identify the purpose of an instruction pointer or program counter register
- Identify the difference between sequential execution and non-sequential execution
- Identify how the value of instruction pointer or program counter register value changes
- Identify the difference between conditional and unconditional control statements
- Identify the two basic types of conditional control statements: branching and looping
- Describe branching/selection control statements
- Identify the difference between sequential statement execution and branching control statement execution
- Identify the syntax of an if control statement

- Identify the valid and invalid conditional expressions that can be used in an "if statements
- List all alternative syntaxes for the if statement
- Apply the most appropriate if statements in program writing
- Distinguish block statements and simple statement in program
- Identify the section of the code that the if statement applies to
- Identify the syntax of an switch control statement
- Identify the valid and invalid switch statement expressions
- List the method how to use case statement
- Know the purpose of break statement in case statements
- Apply switch statements in program writing as appropriate
- Identify the section of the code that the switch statement applies to
- Formulate what kind of flowchart can be used to represent if statements and switch statements
- Describe iterative/looping control statement
- Identify the difference between branching control statement execution and looping control statement execution
- Identify the different looping constructs in C++
- Identify counter controlled loop and sentinel based loop
- Identify the C++ control statements appropriate for counter controlled loop
- Identify the C++ control statements appropriate for sentinel based loop
- Identify the syntax of a for loop, while loop and do..while loop control statement in C++
- Identify when a for loop control statement is more appropriate that the others
- Identify when a while loop control statement is more appropriate that the others
- Identify when a do..while loop control statement is more appropriate that the
- Identify the valid and invalid conditional expressions that can be used in these looping control statements
- Apply the most appropriate looping control statements in program writing
- Identify the section of the code that the loop statement applies to
- Formulate what kind of flowchart can be used to represent a for loop, while loop and do..while loop statements in C++

- Identify the role of the break statement in a loop construct
- Identify the role of the continue statement in a loop construct
- Identify the role of the return statement in a loop construct

Self-test Questions

1. Write a program that accept a number and return the type of the number ("even or odd")
2. Write a flowchart that search the smallest prime number greater than or equals to N.
3. Write a program that display sequence of n positive integers starting from a user specified number m
4. Write a program that convert a mark for a course to its corresponding letter-grade using the following scale

Mark	Grade	Mark	Grade
>=90	A+	>=55	C+
>=80	A	>=45	C
>=75	B+	>=30	D
>=60	B	<30	F

5. Write a program that compute and display the sum of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.
6. Write a program that accept two numbers and an operator and compute and display the result of the expression (be careful to handle division by zero)
7. Write a program that computes the number of days in a given month and year of a Gregorian calendar.
8. Write a program that compute and display the next date of a specific date in European calendar accepted from the user. You must consider leap year condition along with other conditions.
9. Write a program that search and display the maximum and the minimum of a list of numbers entered by the user. Where the size of the list is to be entered by the user.
10. Write a program that calculate and display the factorial of a number n where n is a positive number to be entered by the user.
11. Design a flow chart to accept N numbers and display them in reverse order

Questions with answer key

1. Write a program that checks whether a given number is prime or not. A prime number is a number that has only two distinct factors (1 and the number itself)
2. Write a program that compute and display the product of a range of consecutive numbers where the starting and ending numbers of the range are to be entered by the user.
3. Write a program that accept a list of integers different from zero and terminate when the input is zero. The program finally display the number of positive and negative integers
4. Write a program that calculate income tax of an employee given the total monthly salary from the keyboard following the following personal tax rule of Ethiopia and display the tax on the screen
   a. All salary amounts less than or equal to 150 Birr are tax free.
   b. Salary amounts between 151 to 650 birr will be deducted 10% of the salary – 15 birr
   c. Salary amounts between 651 to 1400 birr will be deducted 15% of the salary – 47.50 birr
   d. Salary amounts between 1401 to 2350 birr will be deducted 20% of the salary – 117.50 birr
   e. Salary amounts between 2351 to 3550 birr will be deducted 25% of the salary – 235 birr
   f. Salary amounts between 3551 to 5000 birr will be deducted 30% of the salary – 412.50 birr
   g. Salary amounts above 5000 birr will be deducted 35% of the salary – 662 birr

Answer key

1.

```
#include<iostream.h>
int main(){
    int N;
    cout<<"Enter the value of N - -> "; cin>>N;
    if(N < 2) {cout<<N<<" is not prime "<<endl; return 0; }
    if(N == 2) {cout<<N<<" is prime\n"; return 0; }
    else{
        for(int i=2; i <N; i++){
            if(N % i == 0){cout<<N<<" is not prime\n"; return 0; }
        }
        cout<<N<<" is prime \n"; return 0;
    }
}
```

2.

```
#include<iostream.h>
int main(){
    int product=1, start, end;
    cout<<"Enter the starting and ending number  - -> "; cin>>start>>end;
    for(int i=start; i <=end; i++)product *=i;

    cout<<"The product of integers from "<<start<<" to "<<end<<"  = "<<product<<endl;
    return 0;
}
```

3.

```
#include<iostream.h>
int main(){
    int countPositive=0, countNegative=0, value;
    do{
        cout<<"enter a number 0 to exit "; cin>>value;
        if(value > 0) countPositive++;
        else if(value < 0) countNegative ++;
    }while(value !=0);
    Cout<<"You entered "<<countPositive << " positive numbers and ";
    Cout<<countNegative<<" negative numbers\n";
    return 0;
}
```

4.

```
#include<iostream.h>
int main(){
    float grossSalary, incomeTax;
    cout<<"Please enter the gross salary "; cin>> grossSalary;
    if(grossSalary <= 150) incomeTax=0;
    else if(grossSalary <= 650) incomeTax= grossSalary*0.1 – 15.0;
    else if(grossSalary <= 1400) incomeTax= grossSalary*0.15 – 47.50;
    else if(grossSalary <= 2355) incomeTax= grossSalary*0.2 – 117.50;
    else if(grossSalary <= 3550) incomeTax= grossSalary*0.25 – 235.0;
    else if(grossSalary <= 5000) incomeTax= grossSalary*0.3 – 412.50;
    else incomeTax= grossSalary*0.35 – 662.0;
    cout<<"The employee income tax = "<<incomeTax<<endl;
    return 0;
}
```

Unit Four

Arrays, Strings and Pointers in C++

Summary

Solving problem by computer needs computation of finite set of instructions step by step in a well-controlled fashion. This computation is made to process input data and transform the data into output information. The data which is the primary source of the information must be stored so that computation can be made systematically and efficiently. Usually the data needs to be manipulated is large in size and complex in structure. Array, string and pointers are structures in programming used to represent data for use in computational processing.

General Objective

The purpose of this unit is to introduce student with the array, string and pointer constructs that are vital to represent large and complex data for efficient computational algorithm.

Specific Objectives

At the end of this unit, students should be able to:

- To explain what an array is and how array objects are structured in memory
- To explain why array elements must have the same data type
- To explain why array elements are arranged in contiguous memory space
- Identify how array elements are referred using the array name and how the exact location of the array be calculated
- Describe how array identifier is declared and initialized
- Identify similarities and differences between array element referred by the array name and its index and simple variable identifier with the same type as the array elements
- Explain how array can be passed as an argument to a function
- Identify the difference between array of character and string
- Explain how to use array and string for representing collection of data
- Explain how to initialize array of character with string value
- Identify how array elements are referred using the array name
- Identify some of the most important string manipulation functions and how to use them

- Explain how to convert string into integer, long integer or floating point number
- Distinguish 1-dimensional and multi-dimensional array
- Identify the syntax to declare multi-dimensional array identifier
- Illustrate 2-D array as array of elements where each elements are 1-D array
- Illustrate 3-D array as array of elements where each elements are 2-D array
- Apply multi-dimensional array to represent, access and process complex data
- Explain what a pointer and pointer variables are
- Distinguish the difference between address and value of a variable identifier
- Identify the syntax to declare pointer variable
- Explain why pointer variable declaration require type specification
- Explain what is a NULL pointer value
- Identify how to assign pointer variable a value which is address of compatible variable identifier, element of an array identifier, value of array identifier
- Explain the semantics/meaning of arithmetic on pointer variable identifier
- Identify how to access value of an object using the array variable value which store the address of the object
- Identify how to allocate memory dynamically to the pointer variable
- Explain how to manage dynamically allocated memory using pointer variable
- Identify how to de-allocate memory allocated dynamically and pointed by the pointer variable
- Explain what an array of pointer is
- Explain how to reserve memory, use the memory, and de-allocate the memory assigned for an array of pointer
- Explain about pointer of pointer variable identifier and its declaration
- Identify how to assign memory location dynamically to pointer of pointer variable identifier
- Identify how to de-allocate memory location dynamically assigned to pointer of pointer variable identifier
- Identify why one needs a pointer variable of type void *

Self-test Questions

1. Write a program that stores temperatures for 4 weeks to an array from the user, and identify the week with the highest average temperature (1-week is 7 days).
2. Rewrite question number 2 and sort them in ascending order and display the numbers before and after sorting.
3. Rewrite question number 2 and sort them in descending order and display the numbers before and after sorting.
4. Write a program that counts the number of times an item appears among the elements of a list of integers stored in an array from the keyboard and displays that count as the frequency of the number in the list.
5. Write a program that will reverse the first half elements of an array of integers with the second half elements symmetrically, without the need to declare another array. E.g. An array having the elements {a,b,c,d,e,f} when reversed, it will be {d,e,f,a,b,c}.
6. Write a program that accept N integers and display them in reverse order. Use pointers to avoid over estimation or under estimation of memory reservation)
7. Write a program that tracks the grades for 5 classes, each having 10 students. Input the data through keyboard. Print the table in its native row and column format.
8. Modify the above question assuming that the number of students in each class is not known. (Hint: use pointer of array)
9. Write a program that merges two sorted integer arrays in a third array. The merged array should be sorted too.
10. Write a program that stores and prints the numbers from 1 to 21 in a 3-by-7 table.
11. Write a program that accepts a square matrix (NxN dimension where N is unknown) and displays the summation of:
    a. the rows separately
    b. the columns separately and
    c. the two diagonals

    (Use pointer of pointers to properly manage the required memory space to hold the matrix)

    10. Write a program that checks whether a word is palindrome (read the same forwards and backwards) or not.

11. Write and test a program that performs perfect shuffle on an array of integers. A perfect shuffle is a sequence obtained by interleaving its first half with the second half, always moving the middle card to the front. For example, the perfect shuffle of {1,2,3,4,5,6,7,8,9,10} is {6,1,7,2,8,3,9,4,10,5}
12. Write a program that reads two 4x4 square matrices, A and B, and then displays
    a. A+B
    b. A-B and
    c. A*B
13. Extend the above question so that dimension is NxN and apply concept of pointer
14. Write and test a program that transposes a square matrix.
15. Write a program to find the number of times that a given word occurs in a sentence

    Interaction with the program might look like:

    The word is "the".

    The sentence is "the cat sat on the mat".

    The word occurs 2 times.
16. SMUC plan to have a system that keep daily track of students ID that enter into the campus. A student ID is 14 characters long. Initially N (say N=1500) byte long buffer of character is prepared to keep N/15 (100) students ID. The user keeps entering students' ID when they enter into the campus. However, after a while, the buffer may be full and require creating a new buffer of size 2N and transferring the old buffer data into the new buffer and the old buffer size should be reused. Again if the new buffer get full, we should create 3N sized buffer, 4N sided buffer and so on. Write the program that successfully implements this idea.

Questions with answer key

1. Write a program that Search a searchKey from an array of N integers constructed by getting its value from a file (call the function **getData** with the array and its MAX size as its arguments. The function will read the data and store into the array and return the number of integers stored if successful; 0 if the data doesn't exist or size exceeds the specified size or any other problem exist). The program should return the index of the array if it founds the searchKey or -1 to mean, it doesn't exist.
2. Write a program that accept N integers from the user and print the maximum, the minimum, average and standard deviation of the data values.
3. Write a program that stores daily sales price data of a company XYZ which sells computer spare parts. XYZ operate every day including Sunday for the last 2 years. There are also different types of items available in the Company shops. You are asked to define appropriate pointer that can store the amount of birr sold for a given item and a given day. (Note: item and day can be easily represented by integer). For example the first item and in the second day can be represented by the integer 0, 1. Finally display the total sales of each item in all days.

Answer key

1.

```
#include<iostrea.h>
#define MAX 1000
int main(){
        int data[MAX], key, size;
        size = getData(data, MAX);
        if(size == 0){
                cout<<"unable to get the data "<<endl;
                return 1;
        }
        cout<<"Enter your integer to search "; cin>>key;
        int i=0;
        while(data[i++] !=key && size > i);
        if(i < size) cout<<"I found the ket at index "<<i<<endl;
        else cout<<"The key doesn't exist in the list"<<endl;
        return 0;
}
```

2.

```
#include<iostrea.h>
int main(){
    int *data, min,max,size, sum=0;
    float average, stdv=0;
    cout<<"How many integers you need to store ==>"; cin>>size;
    data = new int[size];
    if(size == 0) return 0;
    cout<<"Enter data[0] = "; cin>>data[0];
    max = min = data[0];
    for(int i=1; i < size; i++){
            cout<<"Enter data["<<i<<"] = "; cin>>data[i];
            sum +=data[i];
            if(min > data[i]) min = data[i];
            if(max < data[i]) max = data[i];
    }
    average = float(sum)/float(size);
    for(int i=0; i < size; i++)
            stdv +=(data[i]- average)* (data[i]- average);
    stdv = stdv/size;
    cout<<"Distribution of the data\n";
    cout<<"Min Value        \t\t"<<min<<endl;
    cout<<"Max Value        \t\t"<<max<<endl;
    cout<<"Average Value\t\t"<<Average<<endl;
    cout<<"Standard Deviation\t"<<stdv<<endl;
    return 0;
}
```

3.

```
#include<iostrea.h>
int main(){
    float **data;
    int rows, cols;
    cout<<"How many days of data you need to store==>"; cin>>rows;
    cout<<"How many items of data you need to store==>"; cin>>cols;
    data = new (float *) [rows];
    for(int i=0; i<rows; i++){
        data[i] = new float[cols];
         cout<<"enter the data for day="<<i+1;
        for(int j=0; j<col; j++){
            cout<<"   for item="<<j+1<<"  ";
            cin>>data[i][j];
        }
    }
    cout<<"Total sales amount of each item\n";
    float sum;
    for(int i=0; i<cols; i++){
        sum = 0;
        for(int j=0; j<rows;j++) sum +=data[j][i];
        cout<<"Sales for item "<<i+1<<" = "<<sum<<endl;
    }
    for(int i=0; i<rows; i++)
        delete [] data[i];
    delete [] data;
    return 0;
}
```

Unit Five

C++ Functions

Summary

In order to solve complex problem by computer to achieve a desired objective, there are smaller problems needs to be solved before we solve the bigger problem. This the common approach while solving any real world problem. For example, to start some business, getting license, renting building for the business, setting up the environment, purchasing basic assets, employing staffs, cost estimation for service are considered as sub problems. Similarly, computer based solution implementation require identification of smaller problems and solving them independently and use them to solve the main problem. The smaller problems can be independently solved and their result can be communicated to other smaller problems or more complex problems. This unit tries to provide students a direction to address such issues to solve more complex problems. This is called modulirazation. Modularization facilitate team works, debugging solution from logical or run time error, maintenance of software, etc

General Objective

The purpose of this unit is to introduce students the concepts of modularization, which is vital to solve complex problems.

Specific Objectives

At the end of this unit, students should be able to:

- Explain what modularization is
- Explain stepwise bottom-up modularization and stepwise top-down modularization
- Define what a function is
- Explain how function is declared and defined
- Explain why function in C++ needs to be declared
- Distinguish the difference between declaration and definition of functions
- Describe the concept of reusing code/modules/function
- Identify and explain signature of a function
- Explain about arguments of a function
- Explain about parameter passing techniques in function implementation

- Identify parameter passing by value, reference, by address
- Explain how array/pointer can be passed as a parameter (one or more dimension)
- Distinguish the difference between formal and actual arguments
- Distinguish between functions whose return type is **void**, ***(void *)*** and other type
- Distinguish between functions whose argument is ***(void *)*** or **(type *)** for some other type
- Explain how to call a function from another function
- Distinguish the difference between recursive and non-recursive function call
- Distinguish the difference between global and local variables
- Identify scope of variables declared inside a function
- Distinguish the difference between automatic and static variables
- Explain what an inline function is
- Explain how to implement a function that support default value to its one or more arguments
- Identify what an overloaded function is
- Construct an overloaded functions

Self-test Questions

1. Write a C++ function that swaps the value of arguments so that the arguments value will be swapped upon completion of the function execution.

2. Write a C++ function that takes array of integer and its size from a calling function and displays the sum and average within the function body and doesn't return any value back to the caller.

3. Write a C++ function that takes a vector of real numbers as an argument and returns the norm of the vector using the function name.

4. Write a C++ function that takes a positive integer n as an argument and returns the largest power of two greater than or equal to n.

5. Write a C++ function that takes a positive integer n as an argument and returns 1 if n is prime and 0 otherwise.

6. Write a C++ function that takes a positive integer n as an argument and returns the next prime number greater than n
7. Write a program that compute the N^{th} prime number and display the result back to the user. (Use the above two functions)
8. Write a C++ function that takes a positive integer as input and returns the sum of the digits in its decimal representation. For example, the sum of the digits of 234567 is 27.
9. write a recursive function that takes n as an argument and return the sum of the first n integers
10. Write a C++ function that takes a positive integer as input and returns the equivalent number written from right to left. For example, the equivalent number for 234567 when it is written from right to left is 765432.
11. Write a function *"no* repetition*(…)"* which removes all repetition of characters from a string. Test the function in a suitable main program, which should be able to reproduce the following input/output:

    Type in a string: **This string contains repeated characters**
    The string without repetition is: **This trngcoaepd**
12. Write a C++ function that takes three integer variable (a, b, c) in as separate parameters and rotates the values stored so that value of a goes to b, value of b goes to c and c to a.
13. Write a modular program that solves the tower of Hanoi problem

Questions with answer key

1. Write a C++ function that takes 3 integer numbers as an input argument and returns the min and the max of the numbers as an argument.
2. Write a C++ function that takes a positive integer as input and returns the leading digit in its decimal representation. For example, the leading digit of 234567 is 2.
3. Write a C++ function that takes array of integers and its size and returns the sum and the average to the calling function through its parameters.
4. Given N which is the number of students who registered for 5 similar courses where each course has specific credit hours. Write a program that accept letter grades for each

student and display the all the students semester GPA on the screen. (Use pointers as required to optimally use memory)

Answer key

1.

```
void min_max(int val1, int val2, int val3, int *min, int *max){
        *min = (val1 < val2 && val1 < val3? val1 : (v2 <val3 ? val2 : val3));
        *max = (val1 > val2 && val1 > val3? val1 : (v2  > val3 ? val2 : val3));
}
```

2.

```
int leadingDigit(int value){
        int divisor = 1;
        float fValue = value;
        while (fValue / divisor >= 1) divisor *=10;
        return value/divisor;
}
```

3.

```
void statistics(int data[], int size, int *sum, float *average){
        int temp = 0;
        for(int i=0; i<size; i++) temp = temp+data[i];
        *sum = temp;
        *average = (float(temp)/size);
}
```

4.

```
#include<iostream.h>
void getGrade(int N, grade);
void displaySGPA(float * SGPA, int N){
        cout<<"Semester GPA of all students "<<endl;
        cout<<"Student No\t\tSGPA"<<endl;
        for(int i=0; i<N; i++)
                cout<<i+1<<"\t\t"<<SGPA[i]<<endl;
}
float * getSGPA(int N, int ** grade,int  total_credit){
        float * sgpa = new float[N];
        for(int i=0; i < N; i++){
                sgpa[i] = (grade[0][i] + grade[1][i] + grade[2][i] + grade[3][i] + grade[4][i]);
                sgpa[i] /=total_credit;
        }
        return sgpa;
}
int main(){
        int * grades[5]; //grades of the 5 courses for each students
        float *SGPA;
        int N, credit[5] = {3,3,3,3,3}; //assume all courses have three credit but can be changed
        cout<<"How many students you have -->"; cin>>N;
        getGrade(N, grade);
        int total_credit = credit[0]+ credit[1]+ credit[2]+ credit[3]+ credit[4];
        SGPA = getSGPA(N, grade, total_credit);
        displaySGPA(SGPA, N);
}
void getGrade(int N, grade){
        for(int i=0; i <5; i++){
                grades[i] = new int[N];
                cout<<"Enter grades in capital letter of all the students for" cout<<i+1<<endl;
                for(int j=0; j< N; j++){
                  do{
                        int valid = 1;
                        cout<<"Grade of student "<<j+1<<" = [A or B or C or D or F] "; cin>>ch;
                         switch(ch){
                                case 'A': grade[i][j] = A; break;
                                case 'B': grade[i][j] = B; break;
                                case 'C': grade[i][j] = C; break;
                                case 'D': grade[i][j] = D; break;
                                case 'F': grade[i][j] = F; break;
                                default: cout<<"Invalid grade entered"; valid = 0;
                  while(valid==0);
                }
        }
}
```

Unit Six

File Processing in C++ Programming

Summary

While solving problem, input data to the program should be provided to the program and placed in the structured designed to hold data. These data is usually large in size cannot be entered into the system from the keyboard as we did in the previous units. Real problem data may even be used again and again by the same or different application programs. Hence, file input and output is the most appropriate mechanism for most computer based application to be usable. This unit is intended to provide guideline to students so that they will address file processing issue.

General Objective

The purpose of this unit is to introduce student file processing and file management concepts which are vital to solve real world application that demand large input from the environment and output its processing result permanently for future use.

Specific Objectives

At the end of this unit, students should be able to:

- Explain how file can be used as an input device
- Explain how file can be used as an output device
- Explain what stream mean in C++ input/output process
- Distinguish istream and ostream objects in C++
- Distinguish ifstream and ofstream objects in C++
- Identify the limitations of getting input from the keyboard
- Identify the limitations of sending output to the screen
- Explain how to declare input/output file handle for use in C++ program
- Identify the different modes of opening a file
- Distinguish the difference between binary and text file mode
- Explain how to open a file for the specified mode and assign file handle to the appropriate file handle variable

- Use both text and binary file to read its content
- Use both text and binary file to update its content
- Use both text and binary file write into it
- Use the functions that can be used to determine the state of the file opened
- Explain how to seek into a specific location of a file stream
- Explain why one should close a file

Self-test Questions

1. Write a program that accept from a user a line of text and write it back to a file called test.txt under your preferred directory. If the file exist, don't erase the previous content rather append the new text at the end
2. Write a program that reads the content of the file test.txt created in question number 1 and display into the screen
3. Write a program that accept N floating point number from the user and write first N and the N numbers into a file called numers.txt under your preferred directory. If the file exist, respond for the user and stop reading and writing the numbers
4. Repeat the above question so that the file is written as a binary file with filename numbers_in_binary.txt
5. Which one demand larger file size (numers.txt or numbers_in_binary.txt)
6. Write a program that reads the file you created in Question number 3 and display the data value into the screen
7. Write a program that reads the file you created in Question number 4 and display the data value into the screen
8. Write a program that accept a text filename from the user through the keyboard and count the frequency of all the characters and display all the distinct characters with frequency greater than zero together with their frequencies.
9. Write a program that takes two file names as argument and returns true (1) if the content of the two files are identically the same, else returns false (0) and write a main function that implements this function
10. Write a program that first takes two file names filename1 and filename2. The program them copies the content of the first file into the second.

11. Write a program that writes into a file matrix.txt the Matrix A and B each contains NxM dimension of matrix of data where each elements are integer. The program should write first N (number of rows) followed by M (number of columns) then the data content of both A and B row by row.
12. Write a program that reads the file content created at question number 11 and display the sum of the two matrices into a new file call sum.txt with the same format as matrix.txt

Questions with answer key

1. Write a program that define array of N character as a buffer (N > 1000) and that accept a line of text whose size is smaller than 100. The line of the string will be added into the buffer as far as the buffer has space to hold the entire characters of the string. Otherwise the buffer should be cleared by writing its content into a file called buffer.txt and clear the buffer for the intended purpose. The user input terminates while he input a zero length string.

Answer key

1.

```
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#define MAX 1000
int main(){
        ofstream outf;
        char buffer[MAX], str[100];
        int bufferlen = 0;
        outf.open("d:\\buffer.txt", ios::out);
        do{
                cout<<Enter your string "; cin.getline(str, 100,'\n');
                len = strlen(str);
                if(bufferlen + len >= MAX){ //clear the buffer
                        outf<<buffer;
                        bufferlen = 0;
                        strcpy(buffer, str);
                }
                else{
                        strcat(buffer, str);
                        bufferlen  +=len;
                }
        }
        outf<<buffer<<endl;
        outf.close();
        return 0;
}
```

Unit Seven
C++ Structure and Class basics

Summary

We have seen that array and pointers gives as a mechanism to structure input data for use in complex computational procedure. These structure permits us to structure data if the elements come from the same data types. However, real world data are highly complex and elements may not have the same type. Moreover, the previous discussion shows both data and operations are interleaved together which expose data to be managed by any part of your program code. Hence, we may need to separate data and its operation to avoid unauthorized access to the data. This unit is intended to provide guideline to students so that they will address how such complex data can be structured in more appropriate and natural fashion.

General Objective

The purpose of this unit is to introduce student about structures and class basics in C++ which are vital for structuring complex format of data elements consists of various types of basic data elements and with appropriate data protection from being accessed through unauthorized module of the program.

Specific Objectives

At the end of this unit, students should be able to:

- Identify enumerated type in C++
- Use enumerated type in C++ program
- Explain how to define structure in C++
- Explain how a C++ structure memory is arranged
- Explain how to define variable identifier from a structure defined
- Explain how to use structure in declaration statement, parameter passing, dynamic memory allocation and where ever we need data type to be specified
- Identify how to identify an element from a structure variable
- Identify the concept of class in C++
- Explain how to define class in C++
- Explain how a C++ class memory is arranged

- Explain how to define variable identifier from a class (called object)
- Distinguish structure and class in C++ program
- Distinguish Class and Object in C++
- Distinguish the different access modifiers in Object Oriented Programming
- Explain what a constructor and destructor mean in C++ class definition
- Explain how to use class in declaration statement and parameter passing

Self-test Questions

1. What is a structure? Describe the role of a structure to model real world data for programming
2. What is class? Describe the role of class to model real world object
3. What member variables in a structure?
4. What is member variable and member function in a class definition
5. What is data hiding in Object Oriented Programming
6. What is member function in Object Oriented Programming
7. What are private and public access modifiers in Object Oriented Programming
8. Define a structure called Point that define a point in 3D space
9. Define a structure called Vector that define a vector of integer values and its dimension
10. Write a program that accept dimension of two vectors and their elements value from the user so that the program will generate the sum of the two vectors (use the structure defined above)

Questions with answer key

1. Create a class called **VectorClass** that define a vector of floating point with the specified dimension that supports the following operation:
   - Vector Addition
   - Vector difference
   - Dot product of vectors

   The class should have:
   - A constructor that accept dimension and set all its elements value 0.

- A constructor that accept dimension and an array of floating point values and initialize its elements with the array values
- A destructor that release the memory space dynamically allocated for the vector

2. Write a program that implements the VectorClass to Read two points from the keyboard and display their sum, Difference and Dot Product.

Answer key

1.

Class definition

```
#include<iostream.h>
class vector{
        private:
        float *Vector;
        int dimension;
        public:
        vector(int n);
        vector(int n, float *data);
        ~vector();
        int getDimension();
        float *getVector();
        vector vector::addition(vector *v);
        vector vector::difference(vector *v);
        vector vector::multiplication(vector *v);
        float vector::dotproduct(vector *v);
        void display();
};
```

Constructors and destructor

```
vector::vector(int n){
        dimension = n;
        Vector = new float[dimension];
        for(int i=0; i < dimension; i++) Vector[i] = 0.0;
}
vector::vector(int n, float *data){
        dimension = n;
        Vector = new float[dimension];
        for(int i=0; i < dimension; i++) Vector[i] = data[i];
}
vector::~vector(){
        delete [] Vector;
}
```

Accessor

```
int vector::getDimension(){
        return dimension;
}
float * vector::getVector(){
        float *data = new float[dimension];
        for(int i = 0; i < dimension; i++) data[i] = Vector[i];
        return data;
}
```

Other member functions

```
void vector::display(){
        for(int i = 0; i < dimension; i++)
                cout<<Vector[i]<<"  ";
        cout<<endl;
}
vector vector::addition(vector *v){
        float *data = v->getVector();
        for(int i=0; i < dimension; i++)
                data[i] = data[i] + Vector[i];
        vector *result = new vector(dimension, data);
        delete[] data;
        return (*result);
}
vector vector::difference(vector *v){
        float *data = v->getVector();
        for(int i=0; i < dimension; i++) data[i] =Vector[i] - data[i];
        vector *result = new vector(dimension, data);
        delete[] data;
        return (*result);
}
vector vector::multiplication(vector *v){
        float *data = v->getVector();
        for(int i=0; i < dimension; i++) data[i] =Vector[i] * data[i];
        vector *result = new vector(dimension, data);
        delete[] data;
        return (*result);
}
float vector::dotproduct(vector *v){
        float sum = 0.0;
        float *data = v->getVector();
        for(int i=0; i < dimension; i++) sum += Vector[i] * data[i];
        delete[] data;
        return sum;
}
```

```
2
int main(){
        int dim;
        cout<<"Please enter the dimension of the two vectors "; cin>>dim;
        float * data = new float[dim];
        cout<<"Please enter the first vector "<<endl;
        for(int i = 0; i < dim; i++)
                cin>>data[i];

        vector *v1 = new vector(dim, data);
        v1->display();
        cout<<"please enter the second vector "<<endl;
        for(int i = 0; i < dim; i++)
                cin>>data[i];

        vector *v2 = new vector(dim, data);
        v2->display();

        vector v3 = v1->addition(v2);
        cout<<"The sum of the vector is \n\t\t";
        v3.display();
        vector v4 = v1->difference(v2);
        cout<<"The difference of the vector is \n\t\t";
        v4.display();
        vector v5 = v1->multiplication(v2);
        cout<<"The product of the vector is \n\t\t";
        v5.display();
        float v6 = v1->dotproduct(v2);
        cout<<"The dot product of the vector is \n\t\t"<<v6<<endl;
        return 0;
}
```