SOFTWARE PRACTICES 1

Lecture Notes_2

ALGORITHM DEVELOPMENT

Algorithm

An algorithm is an effective step-by-step procedure for solving a problem in a finite number of steps. In other words, it is a finite set of well-defined instructions or step-by-step description of the procedure written in human readable language for solving a given problem.

Algorithm outlining the operation of a computer program, written in something similar to computer language but in a more understandable format. In other to be able to solve a particular problem one must know the steps or routes to the solution of that problem.

Elements of an Algorithm

The basic elements of an algorithm are sequence, selection, and iteration.

- **Sequence** - the order in which behaviors and commands are combined in a project in order to produce a desired result.
- **Selection** - is the use of conditional statements in a project. Conditional statements such as [If then], or [If then else] affect the project flow of a VEXcode VR project.
- **Iteration** - algorithms often use repetition to execute steps a certain number of times, or until a certain condition is met. This is also known as "looping." Iteration can change the project flow by repeating a behavior a specified number of times or until a condition is met.

Characteristics/Properties of an Algorithm

Not all procedures can be called an algorithm. An algorithm should have the following characteristics

- **Unambiguous** − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- **Input** − An algorithm should have 0 or more well-defined inputs.
- **Output** − An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- **Finiteness** − Algorithms must terminate after a finite number of steps.
- **Feasibility** − Should be feasible with the available resources.
- **Independent** − An algorithm should have step-by-step directions, which should be independent of any programming code.

Algorithm Efficiency

Two areas are important for performance:
Space efficiency - the memory required, also called, space complexity
Time efficiency - the time required, also called time complexity

Space Efficiency

There are some circumstances where the space/memory used must be analyzed. For example, for large quantities of data or for embedded systems programming.
Components of space/memory use:

1. instruction space	Affected by: the compiler, compiler options, target computer (cpu)
2. data space	Affected by: the data size/dynamically allocated memory, static program variables,
3. run-time stack space	Affected by: the compiler, run-time function calls and recursion, local variables, parameters

Time Efficiency

Clearly the more quickly a program/function accomplishes its task the better.
The actual running time depends on many factors:

- The speed of the computer: cpu (not just clock speed), I/O, etc.
- The compiler, compiler options .
- The quantity of data - ex. search a long list or short.
- The actual data - ex. in the sequential search if the name is first or last.

For any time complexity consideration there are 3 cases:

- worst case
- average case
- best case

Usually worst case is considered since it gives an upper bound on the performance that can be expected. Additionally, the average case is usually harder to determine (its usually more data dependent), and is often the same as the worst case. Under some circumstances it may be necessary to analyze all 3 cases (algorithms topic).

Algorithm design techniques

Following are some of the main algorithm design techniques:

- Brute-force or exhaustive search.
- Divide and Conquer.
- Greedy Algorithms.
- Dynamic Programming.

- Branch and Bound Algorithm.
- Randomized Algorithm.
- Backtracking.

Problem Solving consists of the following technical Activities

1. Algorithm development,
2. Flowcharting or Pseudocoding,
3. Coding in a particular programming language.

Algorithm development

Algorithm development is the act of designing the steps that solve a particular problem for a computer or any other device to follow not excluding human being, but in this case computers only and computer like devices.

Steps in development of Algorithms

Every problem solution starts with a plan. That plan is called an algorithm.

There are many ways to write an algorithm. Some are very informal, some are quite formal and mathematical in nature, and some are quite graphical. The instructions for connecting a DVD player to a television are an algorithm. A mathematical formula such as πR^2 is a special case of an algorithm. The form is not particularly important as long as it provides a good way to describe and check the logic of the plan.

The development of an algorithm (a plan) is a key step in solving a problem. Once we have an algorithm, we can translate it into a computer program in some programming language.

Algorithm development process consists of five major steps.

Step 1: Obtain a description of the problem.
Step 2: Analyze the problem.
Step 3: Develop a high-level algorithm.
Step 4: Refine the algorithm by adding more detail.
Step 5: Review the algorithm.

Algorithm Representation

An Algorithm can be expressed, developed, written, or represented in one of the following languages or forms:

1. Narrative
2. Flowchart
3. Pseudo code

A flowchart is a diagram of the sequence of operations in a computer program or an accounting system.
A pseudo code is a statements outlining the operation of a computer program, written in something similar to computer language but in a more understandable format.

Standard Methods in solving a Problem
A problem can either be solve using any or both of these methods viz:

1. Top Down Method
2. Bottom Top Method

The Top Down Method starts to solve a problem at the most general level and works toward details or specifics. it is also an approach to a problem that begins at the highest conceptual level and works down to the details. ***While***

The Bottom Top method is an approach to a problem that begins with details and works up to the highest conceptual level.

How to Write an Algorithm?

There are no well-defined standards for writing algorithms. Rather, it is problem and resource dependent. Algorithms are never written to support a particular programming code.

As we know that all programming languages share basic code constructs like loops (do, for, while), flow-control (if-else), etc. These common constructs can be used to write an algorithm.

We write algorithms in a step-by-step manner, but it is not always the case. Algorithm writing is a process and is executed after the problem domain is well-defined. That is, we should know the problem domain, for which we are designing a solution.

Example
Let's try to learn algorithm-writing by using an example.
Problem − Design an algorithm to add two numbers and display the result.
Step 1 − START
Step 2 − declare three integers a, b & c
Step 3 − define values of a & b
Step 4 − add values of a & b
Step 5 − store output of step 4 to c
Step 6 − print c
Step 7 – STOP

Algorithms tell the programmers how to code the program. Alternatively, the algorithm can be written as −

Step 1 − START ADD
Step 2 − get values of a & b
Step 3 − c ← a + b
Step 4 − display c
Step 5 − STOP