CPS 301: STRUCTURED PROGRAMMING

INTRODUCTION TO STRUCTURED PROGRAMMING PARADIGM

Definition

Structured programming is an approach that breaks a program into logical sections called **modules,** in such a way to minimize the program's complexity.

In other words, modularity of program's source code is the essence of structured programming.

Structured Programming

- Structured programming is a way of ensuring that subsidiary sections are arranged in such a way that they can be worked on independently. Each section can be tested and debugged as it is completed; thus considerably enhancing the chances of obtaining a working programme.

- If problems occur, they can be isolated and found very more easily than in a single complete program. U only need to correct the code in the section which the problem occurred, which makes reprogramming to be minimized.

A structured program must be made of series of sections(modules). Each section must be able to do the following:

- It has a label or name
- Perform only one specified task
- Complete the task satisfactorily before continuing
- Have only one entry point
- Have only one exit point
- Be composed of necessary program statements or calls to other subsections, which performs various subtasks and which themselves strictly satisfies these conditions.
- Should only call another module at a lower level than itself
- Be independent

Approaches To Structured Program Design

Program design takes a unique position in any software development process. In program designing , we always look for mechanisms that best implement a solution to the problem we are solving.

There are two approaches normally used to design a structured program:

1. Top-down approach
2. Bottom-up approach

Top-Down Approach

Also called modular design. It encompasses the concept of stepwise refinement. In this approach, the problem is being examined and analyzed from the highest or more general level to a more detailed level. This means that atop-down analysis examines all modules but first translate higher level to a more detailed concrete program specifications before descending to lower levels. At any point, only data and control information with structures necessary for a module are defined. The details of the design at lower levels remain hidden.

In this approach, as little codes as possible need to be written. This allows an error in the design of the program to be corrected before too much work or time has been spent on trying to implement it.

Essential Elements of Top-down Approach

- Design the program in levels or modules
- It must be initial language – dependent.
- Postponement of details to the lower level
- Defining the interface with the lower level
- Verification of each module as it is written.

Advantages Of Top-down Approach

- It enables one to concentrate on solving smaller, more controllable problems in such a way that the solutions probably will fit together to complete the entire programming project.
- Parts of the program may be tested and changed without affecting other parts.
- This approach makes it possible to allocate parts of the large programming project to different programmers.

Bottom-up approach

This approach begins with the lowest levels of the design to the highest, i.e. we work from the detailed to the general. Common routines/functions that may be used by higher modules are identified. For instance, a system having many modules that needs to sort a list. Then sort function is shared commonly by all these modules.

Steps that are taken in Bottom-up Approach

- Each lower level functions or modules are identified as independent tasks to be completed.
- Each function/module is integrated into a program by creating a separate routine.
- Finally, these routines are build from the

Criteria for Judging a Program

These yardsticks are normally used to judge if a program is good or not.

Problem Solvability: does the program solve the problem as requested(i.e. requirements specifications)?

Workability: does the program work under all conditions.?

Sufficient Information: Does it include clear and sufficient information for its users.(i.e. instructions and documentation)?

Logically Written: Is the program logically and clearly written, with short modules and subprograms as appropriate?

Procedure-Oriented and Object -Oriented Programming Approach.

Conventional programming, using high-level languages such as FORTRAN and C , is commonly known as procedure-oriented programming(POP).

In POP approach, the problem is viewed as a sequence of things to be done, such as reading, calculating and printing. A number of functions are written to accomplish these tasks. The primary focus is on functions. POP basically consists of writing a list of instructions for the computer to follow and organizing these instructions into groups known as functions. We normally use a flowchart to organize this actions and represent the flow of control from one action to another.

While we concentrate on the development of functions, very little attention is given to the data that are being used by various functions. What happens to the data? How are they affected by the functions that work on them?

Characteristics of POP

- Emphasis is on doing things(algorithms)
- Large programs are divided into smaller programs known as functions
- Most of the functions share global data
- Data move around the system from function to function
- Functions transform data from one form to another.
- Employs top-down approach in program design.

Drawbacks of POP

1. *Vulnerability of Global Data to inadvertent change by a function*: In a multi-function program, many data items are placed as *global* so that they may be accessed by all the functions. Each function may have its own local data. In large program, it is very difficult to identify what data is used by which function. This provides an *opportunity for bugs to creep in*.
2. Another serious drawbacks is that it doesn't model real world problems very well. This is because functions are action-oriented and do not really correspond to the elements of the problem.

Object-Oriented programming(OOP)

OOP is the most recent concept among programming paradigms. The major motivating factor in the invention of OOP is to remove some of the flaws encountered in the procedural approach. OOP treats data as a critical element in the program development and doesn’t allow it to flow freely around the system. It ties data ore closely to the function that operate on it, and protects it from accidental modification from outside functions.

OOP allows decomposition of a problem into a no of entities called objects and then builds data and functions around these objects. The data of an object can be accessed only by the functions associated with that object. However functions of one object can access the functions of other objects

Features of OOP

- Emphasis is on data rather than procedure
- Programs are divided into what are known as objects
- Data structures are designed such that they characterize the objects
- Functions that operate on the data of an object are tied together in the data structure.
- Data is hidden and cannot be accessed by external functions
- Objects may communicate with each other through functions
- New data and functions can be easily added whenever necessary
- Follows botto-up approach in program design..

Basic Concepts of OOP

Some of the concepts used in OOP includes:

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Message Passing

Introduction to Objects

Objects are keys to understanding object-oriented technology. Objects are just like real world objects we see around. Real- world objects share two characteristics: They all have *state* and *behaviour*. For example, dogs have state(name, colour, breed, hungry) and behaviour(barking, fetching and wagging tail). Students have state (name, matric no, address, mobile no) and behaviour(listening, talking, writing, sleeping).

Software objects are conceptually similar to real-world objects: they too consist of state and related behaviour. An object stores its state in variables (i.e. maintains its state in one or more variables), and exposes(implements) its behaviour through methods (functions, subroutine or procedure in some programming languages). Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

Introduction to Objects(2)

Bundling code into individual software objects provides a number of benefits, including:

- *Modularity:* The source code for an object can be written and maintained independently of the source code for other objects. Once created, an object can be easily passed around inside the system.
- *Information-hiding:* By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- *Code re-use:* If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.
- *Pluggability and debugging ease:* If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement. This is analogous to fixing mechanical problems in the real world. If a bolt breaks, you replace it, not the entire machine.19