Structured programming concepts:

Learning Objectives

- Structures programming
- Modular programming
- Top down analysis
- Bottom up analysis

==========

The concept of Structured programming is dividing the large programme into small modules, so that program become easy to understand.

Even though the Structured Programming Approach has the concept of dividing a program in to multilevel sub programs, but it is considered as a single structure. It means that the code will execute the instruction by instruction one after the other moving from one structure to another structure. It doesn't support the possibility of direct jumping from one instruction to some other. Therefore, the instructions in this approach will be executed in a serial and structured manner. The languages that support Structured programming approach are:

- C
- C++
- Java
- C#

==========

Therefore structured program consists of well-structured and separated modules. The entry and exit in a structured program is a single-time event. It means that the program uses single-entry and single-exit elements. Therefore a structured program is well maintained, neat and clean program. This is the reason why the Structured Programming Approach is well accepted in the programming world.

Structured programming (sometimes known as *modular programming*) is a subset of procedural programming that enforces a logical structure on the program being written to make it more efficient and easier to understand and modify.

==========

Structured programming frequently employs a top-down design model, in which developers map out the overall program structure into separate subsections. A defined function or set of similar functions is coded in a separate module or submodule, which means that code can be loaded into memory more efficiently and that modules can be reused in other programs. After a module has been tested individually, it is then integrated with other modules into the overall program structure.

Therefore, Structured programming is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow constructs of selection (if/then/else) and repetition (while and for), block structures, and subroutines.

==========

Advantages of Structured Programming Approach:

1. Easier to read and understand
2. User Friendly
3. Easier to Maintain
4. Mainly problem based instead of being machine based
5. Development is easier as it requires less effort and time
6. Easier to Debug
7. Machine-Independent, mostly.

Disadvantages of Structured Programming Approach:

1. Since it is Machine-Independent, So it takes time to convert into machine code.
2. The program depends upon changeable factors like data-types. Therefore it needs to be updated with the need on the go.
3. Usually the development in this approach takes longer time as it is language-dependent.

==========

Modular programming

Modular programming is the process of subdividing a computer program into separate sub-programs. A module is a separate software component. It can often be used in a variety of applications and functions with other components of the system.

A program may also contain various functions, and these functions, or modules, are sets of statements that perform an operation or compute a value

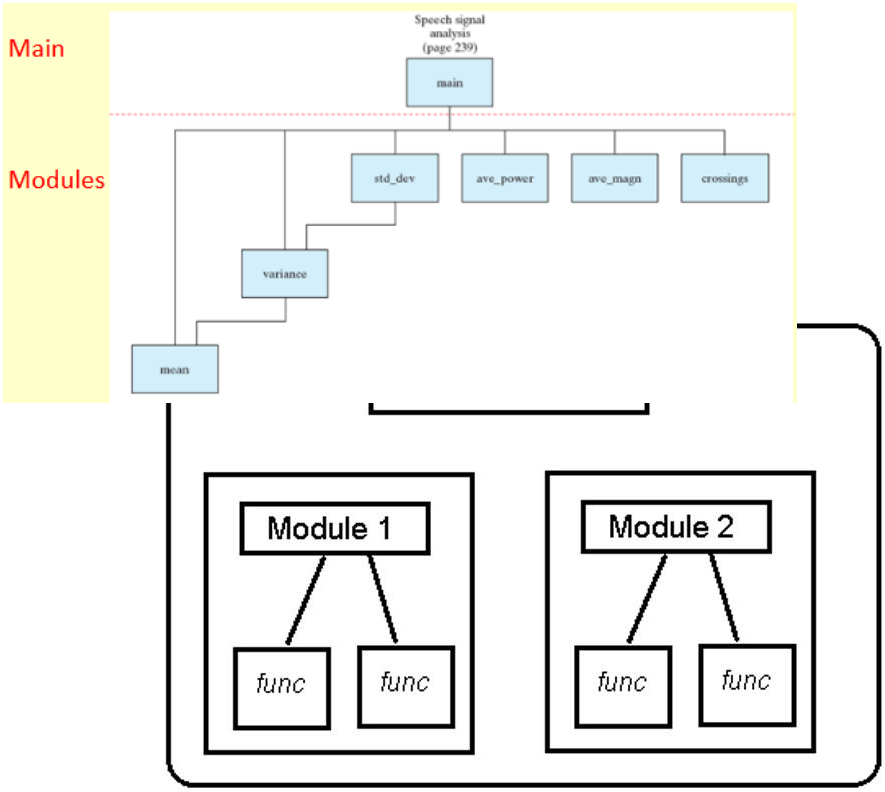
A long and complex program can be divided into smaller subprograms to maintain simplicity and readability.

By separating a solution into a group of modules, each module is easier to understand, thus adhering to the basic guidelines of structured programming

==========

Breaking a problem into a set of modules has many advantages:

- Every module can be written and tested separately from the rest of the program
- A module is smaller than a complete program, so testing is easier
- Once a module has been tested, it can be used in new program without having to retest it (reusability)
- Use of modules (modularity) usually reduces the overall length of programs
- Several programmers can work on the same project if it is separated into modules



The Importance of Modular Programming

Ability to Reuse Code

Modularity enables programmers to use the same code in the module over and over again by simply referencing the code to perform the specific action in the module at different locations of the program. Creating units or classes ensures this.

Easy Debugging

Easy debugging feature can be beneficial for your business.

A program with millions of lines of code will present a huge task if not in a module when debugging is required. You can imagine what it is like to search through such a huge environment to look for errors in a program. Having each task in its discrete module makes the process a lot easier to carry out. A faulty function can easily be checked for errors when in a module.

Simplicity and Ease of Maintenance

It’s like plug and play, efficient!

With modularity, the number of lines of codes should be reduced. Apart from handling the repetition of similar codes, modularity simplifies the whole process of developing a large project by having different programmers work on various aspects of the project.

Organization

Modularization makes reading a program a lot better to do and understand. This makes the programmer better organized in his coding which helps when referencing. This also makes other programmers who may want to go through the workings of the module find it much easier to read. Modularity thus improves readability through a well-organized design.

What Modular Programming Does

Modularity in programming encourages the separation of the functionality in a program into distinct and independent units such that every unit has everything required for it to execute. The elements in the interface of a module can be detected by other modules of the program. Therefore it deals with high-level decomposition of an entire program's code into functional and independent units.

Where is modular programming used?

Modular programming is used where multiple programmers work in a single large project and this enables multiple **programmers** to divide up the work and debug pieces of the **program** independently. Modules in **modular programming** enforce logical boundaries between components and improve maintainability. They are incorporated through interfaces.

Why is modularity important in programming?

Because it encourages the separation of the functionality in a **program** into distinct and independent units for smooth functioning.

The benefits of modular programming are:

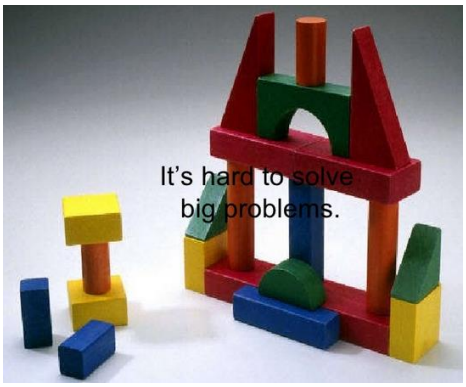
Efficient Program Development. Programs can be developed more quickly with the modular approach since small subprograms are easier to understand, design, and test than large programs. With the module inputs and outputs defined, the programmer can supply the needed input and verify the correctness of the module by examining the output.

Multiple Use of Subprograms. Code written for one program is often useful in others. Modular programming allows these sections to be saved for future use. Because the code is relocatable, saved modules can be linked to any program. With monolithic programming, such sections of code are buried inside the program and are not so available for use by other programs.

Ease of Debugging and Modifying: Modular programs are generally easier to debug than monolithic programs or a single large program. Because of the well-defined module interfaces of the program, problems can be isolated to specific modules. Once the faulty module has been identified, fixing the problem is considerably simpler.

==========

Top down analysis



It is easier to solve small problems that to solve big ones

Computer Programmers use a divide and conquer approach to problem solving

- **a problem is broken into parts**
- **those parts are solved individually**
- **the smaller solutions are assembled into a big solution**

These tecniques are known as top-down design and modular development.

==========

Top-down design is the process of designing a solution to a problem by systematically breaking a problem into smaller, or manageable parts.

==========

First start with a clear statement of problem or concept

a single big idea

==========

next, break it down into several parts.

==========

If any of those parts can be further broken down, then the process continues

==========

and so on…

==========

The final design might look something like this organizational chart, showing the overall structure of separate units that form a single complex entity.

A organizational chart is like an upside down tree, with nodes representing each process.

The leaf nodes are those at the end of each branch of the tree.

==========

The top-down approach goes from the general to the specific.

Top down analysis is a problem solving mechanism in which a given problem is successively broken down into smaller and smaller sub-problems or operations until a set of easily solvable (by computer) sub-problems is arrived at.

These are the levels and each level is numbered commencing with the top (first) level followed by the second level and so on.

Each level marks a different level of abstraction (degree of detail) with the top level exhibiting the greatest degree of abstraction.

Using the top-down approach it is possible to achieve a very detailed breakdown, however, it should be remembered that our aim is to identify easily solvable sub-problems. Our aim is not to use the technique to identify a detailed design or a set of implementation statements.

A detailed design with top down approach is referred to as *top down design*.

This the technique provides a *step wise refinement*.

==========

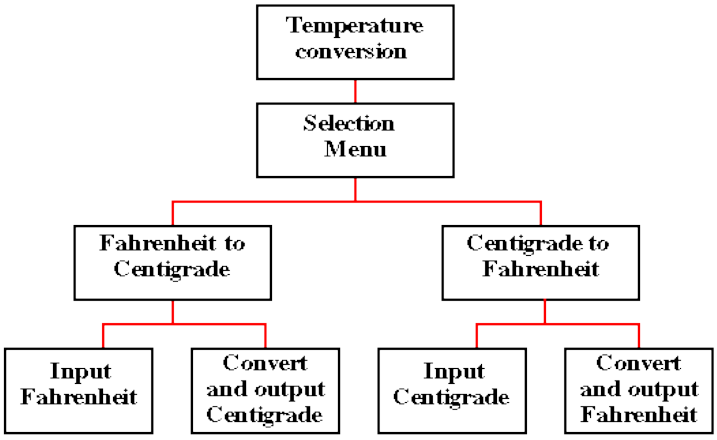
Advantages

- Breaking the problem into parts helps us to clarify what needs to be done.
- At each step of refinement the new parts become leass complicated and therefore easier to figure out
- Parts of the solution may turn out to be reusable.
- Breaking the problem into parts allows more than one person to work on the solution.

The disadvantages are:

1. The designer ends up with very large tree structures.
2. There is no clear identification of the flow of control.
3. It is difficult to represent constructs such as selection, repetition and routine invocation.

Example: Design a top-down analysis for temperature conversion



This is how a programme can be divided into modules and process individually.

Bottom up analysis

Bottom-Up Design Model:

The bottom-up approach begins at the specific and moves to the general.

The specific are the leaf node which is at the bottom and as move upwards it starts becoming to general.

==========

Therefore, in this design, individual parts of the system are specified in details. Then the parts are the linked or combined or integrated to form larger components, which are in turn linked until a complete system is formed. Object oriented language such as C++ or java uses bottom up approach where each object is identified first.

Therefore we solve smaller problems and integrate it as whole and complete the solution.

==========

Advantage:

- Make decisions about reusable low level utilities then decide how there will be put together to create high level construct.

- If the low level modules are tested and then effective integration with other modules can be done.
- Test conditions are easier to create.
- Observation of test results is easier.

Disadvantages

- Sometimes it is difficult to observe system level functions from a partly integrated system. They cannot observe the system level functions until the top level test driver is in place.
- The program as an entity does not exist until the last module is added.

S.No.	TOP DOWN APPROACH	BOTTOM UP APPROACH
1.	In this approach We focus on breaking up the problem into smaller parts.	In bottom up approach, we solve smaller problems and integrate it as whole and complete the solution.
2.	Mainly used by structured programming language such as COBOL, Fortan, C etc.	Mainly used by object oriented programming language such as C++, C#, Python.
3.	Each part is programmed separately therefore contain redundancy.	Redundancy is minimized by using data encapsulation and data hiding.
4.	In this the communications is less among modules.	In this module must have communication.

5.	**It is used in debugging, module documentation, etc.**	**It is basically used in testing.**
6.	**In top down approach, decomposition takes place.**	**In bottom up approach composition takes place.**
7.	**In this top function of system might be hard to identify.**	**In this sometimes we cannot build a program from the piece we have started.**
8.	**In this implementation details may differ.**	**This is not natural for people to assemble.**