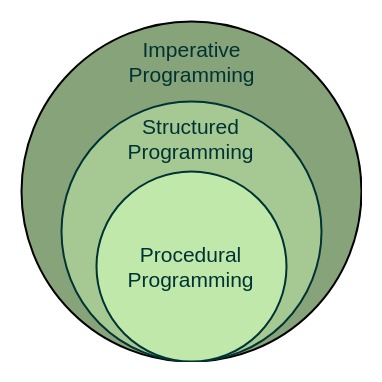
Structured Programming Approach with Advantages and Disadvantages

**Structured Programming Approach**, as the word suggests, can be defined as a programming approach in which the program is made as a single structure. It means that the code will execute the instruction by instruction one after the other. It doesn’t support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. 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#

..etc



The structured program mainly consists of three types of elements:

* Selection Statements
* Sequence Statements
* Iteration Statements

The structured program consists of well structured and separated modules. But 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.

**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 converted machine code is not the same as for assembly language.
3. The program depends upon changeable factors like data-types. Therefore it needs to be updated with the need on the go.
4. Usually the development in this approach takes longer time as it is language-dependent. Whereas in the case of assembly language, the development takes lesser time as it is fixed for the machine.

## What is Program Design?

**Program design** is the process of converting a set of requirements into a collection of commands or a program that can be executed on a computer system. A **program** is a series of instructions that the computer executes in order to perform some meaningful work. For example, the developers at Microsoft created a program called Word that performs word processing activities for a user. Program design is an integral part of software development and depending on the methodology used, can be a significant step in the process. Generally, it consists of:

* Translating requirements into general operations or commands.
* Identifying important attributes and characteristics of the commands and grouping them.
* Establishing an order and relationships between the groups.

The result is a framework that the program can use to deliver the required functionality.

## Top Down Program Design

**Top down** program design is an approach to program design that starts with the general concept and repeatedly breaks it down into its component parts. In other words, it starts with the abstract and continually subdivides it until it reaches the specific. Consider creating the prime factorization of a number like 1540. The steps involved might look like:

* 1540
* 2 x 770
* 2 x 2 x 385
* 2 x 2 x 5 x 77
* 2 x 2 x 5 x 7 x 11

Top down program design works the same way. We start with the overall objective and wind up with a series of steps needed to accomplish it.

## Bottom Up Program Design

**Bottom up** program design works in the exact opposite way. It starts with the component parts and repeatedly combines them to achieve the general concept. In other words, it starts with the specific and continually combines it until it reaches the abstract. For example, consider the factorization from the previous section. For bottom up design the steps involved might look like:

* 2 x 2 x 5 x 7 x 11
* 2 x 2 x 5 x 77
* 2 x 2 x 385
* 2 x 770
* 1540