

Software Development

(Ported from old wiki)

Before going on to discuss the five steps of creating a program it is important to determine what exactly a program is. A program is a list of instructions that contain data for a computer to follow. Different programs are written with different languages. An editing program is made with a different programming "language" than one that uses graphics. Some well known programming languages are COBOL -business, BASIC -language, and - C which is used in science. Programming is complicated process, below is general overview of this 5 step procedure.

Clarify/Define the Problem

Knowing the objective is the first consideration. Is it a payroll or editing program? Knowing who the end user will be is also important. Determining the inputs and outputs is next. How will the program operate and what data is needed to make it happen. After this has been decided feasibility is the next consideration. How many programmers will it take, is the project within budget, does the project have a realistic outline. Finally, if the project is a go, then one must take measures to ensure the project is properly documented and analyzed.

Six mini steps:

- Clarify Objectives & Users
- Clarify Desired Output
- Clarify Desired Input
- Clarify the desired processing
- Double - check feasibility of implementing the program
- Document the analysis

Design the Program

Programs use *algorithms* which are like equations that tell the computer what task to perform. The aim of the programmer is to create algorithms that are clear and simple. Algorithms are expressed first in logical hierarchical form known as *modularization*. Using modules or (a complete thought) the programmer creates a logical thought process for the computer to follow. After that the program is broken down in greater detail using *pseudocode*. Pseudocode uses terms like *if*, *else*, and, *then* to relate the programs rules to the computer.

Two mini steps:

- Determine program logic through top down approach and modularization, using a hierarchy chart
- Design details using pseudocode and/or flowcharts, preferably involving control structure.

A module, a processing step of a program, made up of logically related program statements.

A hierarchy chart, which represents top-down program design, explains the main purpose of the program.
Pseudocode, a way of designing a program which uses normal language statements in order to describe the logic and the processing flow.
Program flowcharts, graphically shows the detailed series of steps

Three control structure:

- Sequence control structure - No decision making
- Selection control structure -
- Loop control structure/Repetition or Iteration Structure
- Case structure

Code the Program

After the program has been designed it must be coded or written. Using the *pseudocode* and logic requirements from step two an appropriate programming language must be selected. As stated in the introduction, coding languages differ in specifications and usability. Once the appropriate code language has been chosen, it is imperative that the programmer follow the *syntax* rules with as little deviation as possible in order for the program to have high accuracy.

Two mini steps:

- Select the appropriate high-level programming language
- Code the program in that language following the syntax carefully

Test the Program

After the program is written it then enters the programming debugging and testing phase of the Program Development Life Cycle (PDLC). During this time the programmer will be looking for errors in both logic and syntax, as well as exploring other areas that may cause the program to either not work properly or to not run at all.

This process is a lengthy and tedious one, oftentimes consisting of up to 50% of a program's time in development (Morley 523). However, with a careful eye paid to program design and coding the amount of time spent debugging can be cut considerably.

As stated, debugging will uncover errors in both logic and syntax. Syntax errors will prevent the program from executing. They can be such simple things as misspelled words or can involve breaking the syntax rules of the programming language used.

On the other hand, logic errors will allow the program to run but will provide incorrect results. Errors of this kind may consist of merely using the wrong relational operator or other, larger, mistakes in writing formulas.

Once the programmer locates the errors they are then fixed and the program is run again. This will happen multiple times, often called "execute, check, and correct" (526), until the program runs flawlessly.

The program will then enter the testing phase.

Reference: Morley, Deborah and Charles S. Parker. Understanding Computers Today and Tomorrow, 13th Edition.

Testing the program comes in two phases, *alpha* and *beta*.

- *Alpha* testing is the process of reading through the program in search of errors in logic. The second step is to run a diagnostic program to search for syntax or input errors.
- *Beta* testing involves using the program in the real world to see if it contains any bugs or other deficiencies.

Document and Maintain

Documentation should be ongoing from the very beginning because it is needed for those involved with program now and future. Upon completion *User Documentation* for commercial use, *Operator Documentation* for people who run computer systems, and *Programmer Documentation* for programmers charged with maintenance.

Four mini steps:

- Write user documentation
- Write operator documentation
- Write programmer documentation
- Maintain the program