1.1. History of Computing and Computers

Definition of a Computer

Before 1935, a computer was a person who performed arithmetic calculations.

Between 1935 and 1945 the definition referred to a machine, rather than a person.

The modern machine definition is based on von Neumann's concepts: a device that accepts input, processes data, stores data, and produces output.

We have gone from the vacuum tube to the transistor, to the microchip. Then the microchip started talking to the modem. Now we exchange text, sound, photos and movies in a digital environment.

Computing milestones and machine evolution:

14th C. - Abacus - an instrument for performing calculations by sliding counters along rods or in grooves.

17th C. - Slide rule - a manual device used for calculation that consists in its simple form of a ruler and a movable middle piece which are graduated with similar logarithmic scales.

1642 – Pascaline -- a mechanical calculator built by Blaise Pascal, a 17th century mathematician, for whom the Pascal computer programming language was named .

1804 - Jacquard loom - a loom programmed with punched cards invented by Joseph Marie Jacquard

1850 - Difference Engine , Analytical Engine - Charles Babbage and Ada Byron. Babbage's description, in 1837, of the Analytical Engine, a hand cranked, mechanical digital computer anticipated virtually every aspect of present-day computers. It wasn't until over a 100 years later that another all purpose computer was conceived.

1939 -1942 - Atanasoff Berry Computer - built at Iowa State by Prof. John V. Atanasoff and graduate student Clifford Berry. Represented several "firsts" in computing, including a binary system of arithmetic, parallel processing, regenerative memory, separation of memory and computing functions, and more. Weighed 750 lbs. and had a memory storage of 3,000 bits (0.4K). Recorded numbers by scorching marks into cards as it worked through a problem.

1940s - Colossus - a vacuum tube computing machine which broke Hitler's codes during WW II. It was instrumental in helping Alan Turing break the German's codes during WW II to turn the tide of the war. In the summer of 1939, a small group of scholars became code breakers, working at Bletchley Part in England. This group of pioneering code breakers helped shorten the war and changed the course of history.

1946 - ENIAC - World's first electronic, large scale, general-purpose computer, built by Mauchly and Eckert, and activated at the University of Pennsylvania in 1946. ENIAC recreated on a modern computer chip. The ENIAC is a 30 ton machine that measured 50 x 30 feet. It contained 19,000 vacuum tubes, 6000 switches, and could add 5,000 numbers in a second, a remarkable accomplishment at the time. A re-programmable machine, the ENIAC performed initial calculations for the H-bomb. It was also used to prepare artillery shell trajectory tables and perform other military and scientific calculations.

Since there was no software to reprogram the computer, people had to rewire it to get it to perform different functions. The human programmers had to read wiring diagrams and know what each switch did. J. Presper Eckert, Jr. and John W. Mauchly drew on Alansoff's work to create the ENIAC, the Electronic Numerical Integrator and Computer.

1951-1959 - vacuum tube based technology. Vacuum Tubes are electronic devices, consisting of a glass or steel vacuum envelope and two or more electrodes between which electrons can move freely. First commercial computers used vacuum tubes: Univac, IBM 701.

1950s -1960s - UNIVAC - "punch card technology" The first commercially successful computer, introduced in 1951 by Remington Rand. Over 40 systems were sold. Its memory was made of mercury filled acoustic delay lines that held 1,000 12 digit numbers. It used magnetic tapes that stored 1MB of data at a density of 128 cpi. UNIVAC became synonymous with computer (for a while).

1960-1968 - transistor based technology. The transistor, invented in 1948, by Dr. John Bardeen, Dr. Walter Brattain, and Dr. William Shockley . It almost completely replaced the vacuum tube because of its reduced cost, weight, and power consumption and its higher reliability. The transistor is made to alter its state from a starting condition of conductivity (switched 'on', full current flow) to a final condition of insulation (switched 'off', no current flow).

1969 - The Internet, originally the ARPAnet (Advanced Research Projects Agency network), began as a military computer network.

1969-1977 - integrated circuits (IC) based technology. The first integrated circuit was demonstrated by Texas Instruments inventor, Jack Kilby, in 1958. It was 7/16" wide and contained two transistors. Examples of early integrated circuit technology: Intel 4004, Dec pdp 8, CRAY 1. Now circuits may contain hundreds of thousands of transistors on a small piece of material, which revolutionized computing.

1976 - CRAY 1 - The world's first electronic digital computer, developed in 1946. A 75MHz, 64-bit machine with a peak speed of 160 megaflops, (one million floating point operations per second) the world's fastest processor at that time.

1976 - Apples/MACs - The Apple was designed by Steve Wozniak and Steve Jobs. Apple was the first to have a "window" type graphical interface and the computer mouse. Like modern computers, early Apples had a peripheral keyboard and mouse, and had a floppy drive that held 3.5" disks. The Macintosh replaced the Apple.

1978 to 1986 - large scale integration (LSI); Alto - early workstation with mouse and Apple. The PC and clone market begins to expand. This begins first mass market of desktop computers.

1990 - Tim Berners-Lee invented the networked hypertext system called the World Wide Web.

1992 - Bill Gates' Microsoft Corp. released Windows 3.1, an operating system that made IBM and IBM-compatible PCs more user-friendly by integrating a graphical user interface into the software. In replacing the old Windows command-line system, however, Microsoft created a program similar to the Macintosh operating system. Apple sued for copyright infringement, but Microsoft prevailed.

Windows 3.1 went to Win 95, then Win 98, Windows XP, Vista, Windows 7/8. There are other OSs, of course, but Windows is the dominant OS today. MACs, by Apple, still have a faithful following. Linux has a faithful following.

1996 - Personal Digital Assistants (such as the Palm Pilot became available to consumers. They can do numeric calculations, play games and music and download information from the Internet.

Pioneer computer scientists

Charles Babbage (1792-1871) - Difference Engine, Analytical Engine. Ada Byron, daughter of the poet, Lord Byron, worked with him. His description, in 1837, of the Analytical Engine, a mechanical digital computer anticipated virtually every aspect of present-day computers.

Alan Turing -- 1912-1954. British Code breaker. Worked on the Colossus (code breaking machine, precursor to the computer) and the ACE (Automatic Computing Engine). Noted for many brilliant ideas, Turing is perhaps best remembered for the concepts of the Turing Test for Artificial Intelligence and the Turing Machine, an abstract model for modeling computer operations. The Turing Test is the "acid test" of true artificial intelligence, as defined by the English scientist Alan Turing. In the 1940s, he said "a machine has artificial intelligence when there is no discernible difference between the conversation generated by the machine and that of an intelligent person." Turing was instrumental in breaking the German enigma code during WWII with his Bombe computing machine. The Enigma is a machine used by the Germans to create encrypted messages.

J. von Neumann -- (1903-1957). A child prodigy in mathematics, authored landmark paper explaining how programs could be stored as data. (Unlike ENIAC, which had to be re-wired to be re-programmed.). Virtually all computers today, from toys to supercomputers costing millions of dollars, are variations on the computer architecture that John von Neumann created on the foundation of the work of Alan Turing's work in the 1940s. It included three components used by most computers today: a CPU; a slow-to-access storage area, like a hard drive; and secondary fast-access memory (RAM). The machine stored instructions as binary values (creating the stored program concept) and executed instructions sequentially - the processor fetched instructions one at a time and processed them. The instruction is analyzed, data is processed, the next instruction is analyzed, etc. Today "***von Neumann architecture***" often refers to the sequential nature of computers based on this model.

John V. Atanasoff -- (1904 - 1995) - one of the contenders, along with Konrad Zuse and H. Edward Roberts and others, as the inventor of the first computer. The limited-function vacuum-tube device had limited capabilities and did not have a central. It was not programmable, but could solve differential equations using binary arithmetic.

J. Presper Eckert, Jr. and John W. Mauchly completed the first programmed general purpose electronic digital computer in 1946. They drew on Alansoff's work to create the ENIAC, the Electronic Numerical Integrator and Computer. In 1973 a patent lawsuit resulted in John V. Atanasoff's being legally declared as the inventor. Though Atanasoff got legal status for his achievement, many historians still give credit to J. Presper Eckert, Jr., and John W. Mauchly the founding fathers of the modern computer. Eckert and Mauchly formed the first computer company in 1946. Eckert received 87 patents.

They introduced the first modern binary computer with the Binary Automatic Computer (BINAC), which stored information on magnetic tape rather than punched cards. Their UNIVAC I ,was built for the U.S. Census Bureau.

Konrad Zuse-- (1910-1995) a German who, during WW II, designed mechanical and electromechanical computers. Zuse's Z1, his contender for the first freely programmable computer, contained all the basic components of a modern computer (control unit, memory, micro sequences, etc.). Zuse, because of the scarcity of material during WW II, used discarded video film as punch cards. Like a modern computer, it was adaptable for different purposes and used on/off switch relays, a binary system of 1s and 0s (on = 1, off = 0). Completed in 1938, it was destroyed in the bombardment of Berlin in WW II, along with the construction plans. In 1986, Zuse reconstructed the Z1.

Early Counting

The start of the modern science that we call "Computer Science" can be traced back to a long ago age where man still dwell-ed in caves or in the forest, and lived in groups for protection and survival from the harsher elements on the Earth. Many of these groups possessed some primitive form of animistic religion; they worshiped the sun, the moon, the trees, or sacred animals. Within the tribal group was one individual to whom fell the responsibility for the tribe's spiritual welfare. It was he or she who decided when to hold both the secret and public religious ceremonies, and interceded with the spirits on behalf of the tribe. In order to correctly hold the ceremonies to ensure good harvest in the fall and fertility in the spring, the shamans needed to be able to count the days or to track the seasons. **From the shamanistic tradition, man developed the first primitive counting mechanisms -- counting notches on sticks or marks on walls.**

From the caves and the forests, man slowly evolved and built structures such as Stonehenge. Stonehenge, which lies 13km north of Salisbury, England, is believed to have been an ancient form of calendar designed to capture the light from the summer solstice in a specific fashion. The solstices have long been special days for various religious groups and cults. Archaeologists and anthropologists today are not quite certain how the structure, believed to have been built about 2800 B.C., came to be erected since the technology required to join together the giant stones and raise them upright seems to be beyond the technological level of the Britons at the time. It is widely believed that the enormous edifice of stone may have been erected by the Druids. Regardless of the identity of the builders, it remains today a monument to **man's intense desire to count and to track the occurrences of the physical world** around him.

Abacus

Meanwhile in Asia, the Chinese were becoming very involved in commerce with the Japanese, Indians, and Koreans. Businessmen needed a way to tally accounts and bills. Somehow, out of this need, the abacus was born. The abacus is the first true precursor to the adding machines and computers which would follow. It worked somewhat like this:

> ***The value assigned to each pebble (or bead, shell, or stick) is determined not by its shape but by its position: one pebble on a particular line or one bead on a particular wire has the value of 1; two together have the value of 2. A pebble on the next line, however, might have the value of 10, and a pebble on the third line would have the value of 100. Therefore, three properly placed pebbles--two with values of 1 and one with the value of 10--could signify 12, and the addition of a fourth pebble with the value of 100 could signify 112, using a place-value notation system with multiples of 10.***

Thus, the abacus works on the principle of place-value notation: the location of the bead determines its value. In this way, relatively few beads are required to depict large numbers. The beads are counted, or given numerical values, by shifting them in one direction. The values are erased (freeing the counters for reuse) by shifting the beads in the other direction. An abacus is really a memory aid for the user making mental calculations, as opposed to the true mechanical calculating machines which were still to come.

Forefathers of Computing

For over a thousand years after the Chinese invented the abacus, not much progress was made to automate counting and mathematics. The Greeks came up with numerous mathematical formulas and theorems, but all of the newly discovered math had to be worked out by hand. A mathematician was often a person who sat in the back room of an establishment with several others and they worked on the same problem. The redundant personnel working on the same problem were there to ensure the correctness of the answer. It could take weeks or months of laborious work by hand to verify the correctness of a proposed theorem. Most of the tables of integrals, logarithms, and trigonometric values were worked out this way, their accuracy unchecked until machines could generate the tables in far less time and with more accuracy than a team of humans could ever hope to achieve.

Blaise Pascal

Blaise Pascal, noted mathematician, thinker, and scientist, built the first mechanical adding machine in 1642 based on a design described by Hero of Alexandria (2AD) to add up the distance a carriage traveled. The basic principle of his calculator is still used today in water meters and modern-day odometers. Instead of having a carriage wheel turn the gear, he made each ten-teeth wheel accessible to be turned directly by a person's hand (later inventors added keys and a crank), with the result that when the wheels were turned in the proper sequences, a series of numbers was entered and a cumulative sum was obtained. The gear train supplied a mechanical answer equal to the answer that is obtained by using arithmetic.

This first mechanical calculator, called the **Pascaline**, had several disadvantages. Although it did offer a substantial improvement over manual calculations, only Pascal himself could repair the device and it cost more than the people it replaced! In addition, the first signs of techno phobia emerged with mathematicians fearing the loss of their jobs due to progress.

Charles Babbage

While Tomas of Colmar was developing the first successful commercial calculator, Charles Babbage realized as early as 1812 that many long computations consisted of operations that were regularly repeated. He theorized that it must be possible to design a calculating machine which could do these operations automatically. He produced a prototype of this "difference engine" by 1822 and with the help of the British government started work on the full machine in 1823. It was intended to be steam-powered; fully automatic, even to the printing of the resulting tables; and commanded by a fixed instruction program.

In 1833, Babbage ceased working on the difference engine because he had a better idea. His new idea was to build an "analytical engine." The analytical engine was a real parallel decimal computer which would operate on words of 50 decimals and was able to store 1000 such numbers. The machine would include a number of built-in operations such as conditional control, which allowed the instructions for the machine to be executed in a specific order rather than in numerical order. The instructions for the machine were to be stored on punched cards, similar to those used on a Jacquard loom.

Herman Hollerith

A step toward automated computation was the introduction of punched cards, which were first successfully used in connection with computing in 1890 by Herman Hollerith working for the U.S. Census Bureau. He developed a device which could automatically read census information which had been punched onto card. Surprisingly, he did not get the idea from the work of Babbage, but rather from watching a train conductor punch tickets. As a result of his invention, reading errors were consequently greatly reduced, work flow was increased, and, more important, stacks of punched cards could be used as an accessible memory store of almost unlimited capacity; furthermore, different problems could be stored on different batches of cards and worked on as needed. Hollerith's tabulator became so successful that he started his own firm to market the device; this company eventually became International Business Machines (IBM).

Konrad Zuse

Hollerith's machine though had limitations. It was strictly limited to tabulation. The punched cards could not be used to direct more complex computations. In 1941, Konrad Zuse, a German who had developed a number of calculating machines, released the first programmable computer designed to solve complex engineering equations. The machine, called the Z3, was controlled by perforated strips of discarded movie film. As well as being controllable by these celluloid strips, it was also the first machine to work on the binary system, as opposed to the more familiar decimal system.

Binary Representation

The binary system is composed of 0s and 1s. A punch card with its two states--a hole or no hole-- was admirably suited to representing things in binary. If a hole was read by the card reader, it was considered to be a 1. If no hole was present in a column, a zero was appended to the current number. The total number of possible numbers can be calculated by putting 2 to the power of the number of bits in the binary number. A bit is simply a single occurrence of a binary number--a 0 or a 1. Thus, if you had a possible binary number of 6 bits, 64 different numbers could be generated. (2^(n-1)) .

Binary representation was going to prove important in the future design of computers which took advantage of a multitude of two-state devices such card readers, electric circuits which could be on or off, and vacuum tubes.

Howard Aiken

By the late 1930s punched-card machine techniques had become so well established and reliable that Howard Aiken, in collaboration with engineers at IBM, undertook construction of a large automatic digital computer based on standard IBM electromechanical parts. Aiken's machine, called the Harvard Mark I, handled 23-decimal-place numbers (words) and could perform all four arithmetic operations; moreover, it had special built-in programs, or subroutines, to handle logarithms and trigonometric functions. The Mark I was originally controlled from pre-punched paper tape without provision for reversal, so that automatic "transfer of control" instructions could not be programmed. Output was by card punch and electric typewriter. Although the Mark I used IBM rotating counter wheels as key components in addition to electromagnetic relays, the machine was classified as a relay computer. It was slow, requiring 3 to 5 seconds for a multiplication, but it was fully automatic and could complete long computations without human intervention. The Harvard Mark I was the first of a series of computers designed and built under Aiken's direction.

Alan Turing

The Turing machine, reads in the symbols from the tape one at a time. What we would like the machine to do is to give us an output of 1 anytime it has read at least 3 ones in a row off of the tape. When there are not at least three ones, then it should output a 0. The reading and outputting can go on infinitely.

Turing's purpose was not to invent a computer, but rather to describe problems which are logically possible to solve. His hypothetical machine, however, foreshadowed certain characteristics of modern computers that would follow. For example, the endless tape could be seen as a form of general purpose internal memory for the machine in that the machine was able to read, write, and erase it--just like modern day RAM.

John W. Mauchly and J. Presper Eckert

Back in America, with the success of Aiken's Harvard Mark-I as the first major American development in the computing race, work was proceeding on the next great breakthrough by the Americans. Their second contribution was the development of the giant **ENIAC (Electrical Numerical Integrator and Computer)** machine by John W. Mauchly and J. Presper Eckert at the University of Pennsylvania. ENIAC used a word of 10 decimal digits instead of binary ones like previous automated calculators/computers. ENIAC also was the first machine to use more than 2,000 vacuum tubes, using nearly 18,000 vacuum tubes. Storage of all those vacuum tubes and the machinery required to keep the cool took up over 167 square meters (1800 square feet) of floor space. Nonetheless, it had punched-card input and output and arithmetically had 1 multiplier, 1 divider-square rooter, and 20 adders employing decimal "ring counters," which served as adders and also as quick-access (0.0002 seconds) read-write register storage.

The executable instructions composing a program were embodied in the separate units of ENIAC, which were plugged together to form a route through the machine for the flow of computations. These connections had to be redone for each different problem, together with presetting function tables and switches. This **"wire-your-own" instruction** technique was inconvenient, and only with some license could ENIAC be considered programmable; it was, however, efficient in handling the particular programs for which it had been designed. ENIAC is generally acknowledged to be the first successful high-speed electronic digital computer (EDC) and was productively used from 1946 to 1955. As controversy developed in 1971, however, over the patentability of ENIAC's basic digital concepts, the claim being made that another U.S. physicist, John V. Atanasoff, had already used the same ideas in a simpler vacuum-tube device he built in the 1930s while at Iowa State College. In 1973, the court found in favor of the company using Atanasoff claim and Atanasoff received the acclaim he rightly deserved.

John von Neumann

In 1945, mathematician John von Neumann undertook a study of computation that demonstrated that a computer could have a simple, fixed structure, yet be able to execute any kind of computation given properly programmed control without the need for hardware modification. Von Neumann contributed a new understanding of how practical fast computers should be organized and built; these ideas, often referred to as the stored-program technique, became fundamental for future generations of high-speed digital computers and were universally adopted. The primary advance was the provision of a special type of machine instruction called conditional control transfer--which permitted the program sequence to be interrupted and re-initiated at any point, similar to the system suggested by Babbage for his analytical engine--and by storing all instruction programs together with data in the same memory unit, so that, when desired, instructions could be arithmetically modified in the same way as data. Thus, data was the same as program.

As a result of these techniques and several others, computing and programming became faster, more flexible, and more efficient, with the instructions in subroutines performing far more computational work. Frequently used subroutines did not have to be reprogrammed for each new problem but could be kept intact in "libraries" and read into memory when needed. Thus, much of a given program could be assembled from the subroutine library. The all-purpose computer memory became the assembly place in which parts of a long computation were stored, worked on piece wise, and assembled to form the final results. As soon as the advantages of these techniques became clear, the techniques became standard practice.

Jack St. Clair Kilby

In 1958, Jack St. Clair Kilby of Texas Instruments manufactured the first integrated circuit or chip. A chip is really a collection of tiny transistors which are connected together when the transistor is manufactured. Thus, the need for soldering together large numbers of transistors was practically nullified; now only connections were needed to other electronic components. In addition to saving space, the speed of the machine was now increased since there was a diminished distance that the electrons had to follow.

For More Detailed Information visit [*http://www.computerhistory.org/timeline/*]

1.2. History of Programming Languages

Ever since the invention of Charles Babbage's difference engine in 1822, computers have required a means of instructing them to perform a specific task. This means is known as a programming language. Computer languages were first composed of a series of steps to wire a particular program; these morphed into a series of steps keyed into the computer and then executed; later these languages acquired advanced features such as logical branching and object orientation. The computer languages of the last fifty years have come in two stages, the first major languages and the second major languages, which are in use today.

In the beginning, Charles Babbage's difference engine could only be made to execute tasks by changing the gears which executed the calculations. Thus, the earliest form of a computer language was physical motion. Eventually, physical motion was replaced by electrical signals when the US Government built the ENIAC in 1942. It followed many of the same principles of Babbage's engine and hence, could only be "programmed" by presetting switches and rewiring the entire system for each new "program" or calculation. This process proved to be very tedious.

In 1945, John Von Neumann was working at the Institute for Advanced Study. He developed two important concepts that directly affected the path of computer programming languages.

The first was known as "shared-program technique". This technique stated that the actual computer hardware should be simple and not need to be hand-wired for each program. Instead, complex instructions should be used to control the simple hardware, allowing it to be reprogrammed much faster.

The second concept was also extremely important to the development of programming languages. Von Neumann called it "conditional control transfer". This idea gave rise to the notion of subroutines, or small blocks of code that could be jumped to in any order, instead of a single set of chronologically ordered steps for the computer to take. The second part of the idea stated that computer code should be able to branch based on logical statements such as IF (expression) THEN, and looped such as with a FOR statement. "Conditional control transfer" gave rise to the idea of "libraries," which are blocks of code that can be reused over and over.

Short Code

In 1949, a few years after Von Neumann's work, the language Short Code appeared. It was the first computer language for electronic devices and it required the programmer to change its statements into 0's and 1's by hand. Still, it was the first step towards the complex languages of today. In 1951, Grace Hopper wrote the first compiler, A-0. A compiler is a program that turns the language's statements into 0's and 1's for the computer to understand. This lead to faster programming, as the programmer no longer had to do the work by hand.

FORTRAN

In 1957, the first of the major languages appeared in the form of FORTRAN. Its name stands for FORmula TRANslating system. The language was designed at IBM for scientific computing. The components were very simple, and provided the programmer with low-level access to the computers innards. Today, this language would be considered restrictive as it only included IF, DO, and GOTO statements, but at the time, these commands were a big step forward. The basic types of data in use today got their start in FORTRAN, these included logical variables (TRUE or FALSE), and integer, real, and double-precision numbers.

COBOL

Though FORTAN was good at handling numbers, it was not so good at handling input and output, which mattered most to business computing. Business computing started to take off in 1959, and because of this, COBOL was developed. It was designed from the ground up as the language for businessmen. Its only data types were numbers and strings of text. It also allowed for these to be grouped into arrays and records, so that data could be tracked and organized better. It is interesting to note that a COBOL program is built in a way similar to an essay, with four or five major sections that build into an elegant whole. COBOL statements also have a very English-like grammar, making it quite easy to learn. All of these features were designed to make it easier for the average business to learn and adopt it.

LISP

In 1958, John McCarthy of MIT created the LISt Processing (or LISP) language. It was designed for Artificial Intelligence (AI) research. Because it was designed for a specialized field, the original release of LISP had a unique syntax: essentially none. Programmers wrote code in parse trees, which are usually a compiler-generated intermediary between higher syntax (such as in C or Java) and lower-level code. Another obvious difference between this language (in original form) and other languages is that the basic and only type of data is the list; in the mid-1960's, LISP acquired other data types. A LISP list is denoted by a sequence of items enclosed by parentheses. LISP programs themselves are written as a set of lists, so that LISP has the unique ability to modify itself, and hence grow on its own.

ALGOL

The Algol language was created by a committee for scientific use in 1958. It's major contribution is being the root of the tree that has led to such languages as Pascal, C, C++, and Java. It was also the first language with a formal grammar, known as Backus-Naar Form or BNF. Though Algol implemented some novel concepts, such as recursive calling of functions, the next version of the language, Algol 68, became bloated and difficult to use. This lead to the adoption of smaller and more compact languages, such as Pascal.

PASCAL

Pascal was begun in 1968 by Niklaus Wirth. Its development was mainly out of necessity for a good teaching tool. In the beginning, the language designers had no hopes for it to enjoy widespread adoption. Instead, they concentrated on developing good tools for teaching such as a debugger and editing system and support for common early microprocessor machines which were in use in teaching institutions.

Pascal was designed in a very orderly approach, it combined many of the best features of the languages in use at the time, COBOL, FORTRAN, and ALGOL. The combination of features, input/output and solid mathematical features, made it a highly successful language. Pascal also improved the "pointer" data type, a very powerful feature of any language that implements it. It also added a CASE statement, that allowed instructions to to branch like a tree.

Pascal also helped the development of dynamic variables, which could be created while a program was being run, through the NEW and DISPOSE commands. However, Pascal did not implement dynamic arrays, or groups of variables, which proved to be needed and led to its downfall. Wirth later created a successor to Pascal, Modula-2, but by the time it appeared, C was gaining popularity and users at a rapid pace.

C Language

C was developed in 1972 by Dennis Ritchie while working at Bell Labs in New Jersey. The transition in usage from the first major languages to the major languages of today occurred with the transition between Pascal and C. Its direct ancestors are B and BCPL, but its similarities to Pascal are quite obvious. All of the features of Pascal, including the new ones such as the CASE statement are available in C. C uses pointers extensively and was built to be fast and powerful at the expense of being hard to read. But because it fixed most of the mistakes Pascal had, it won over former-Pascal users quite rapidly.

Ritchie developed C for the new Unix system being created at the same time. Because of this, C and Unix go hand in hand. Unix gives C such advanced features as dynamic variables, multitasking, interrupt handling, forking, and strong, low-level, input-output. Because of this, C is very commonly used to program operating systems such as Unix, Windows, the MacOS, and Linux.

C++ Language

In the late 1970's and early 1980's, a new programming method was being developed. It was known as Object Oriented Programming, or OOP. Objects are pieces of data that can be packaged and manipulated by the programmer. Bjarne Stroustroup liked this method and developed extensions to C known as "C With Classes." This set of extensions developed into the full-featured language C++, which was released in 1983.

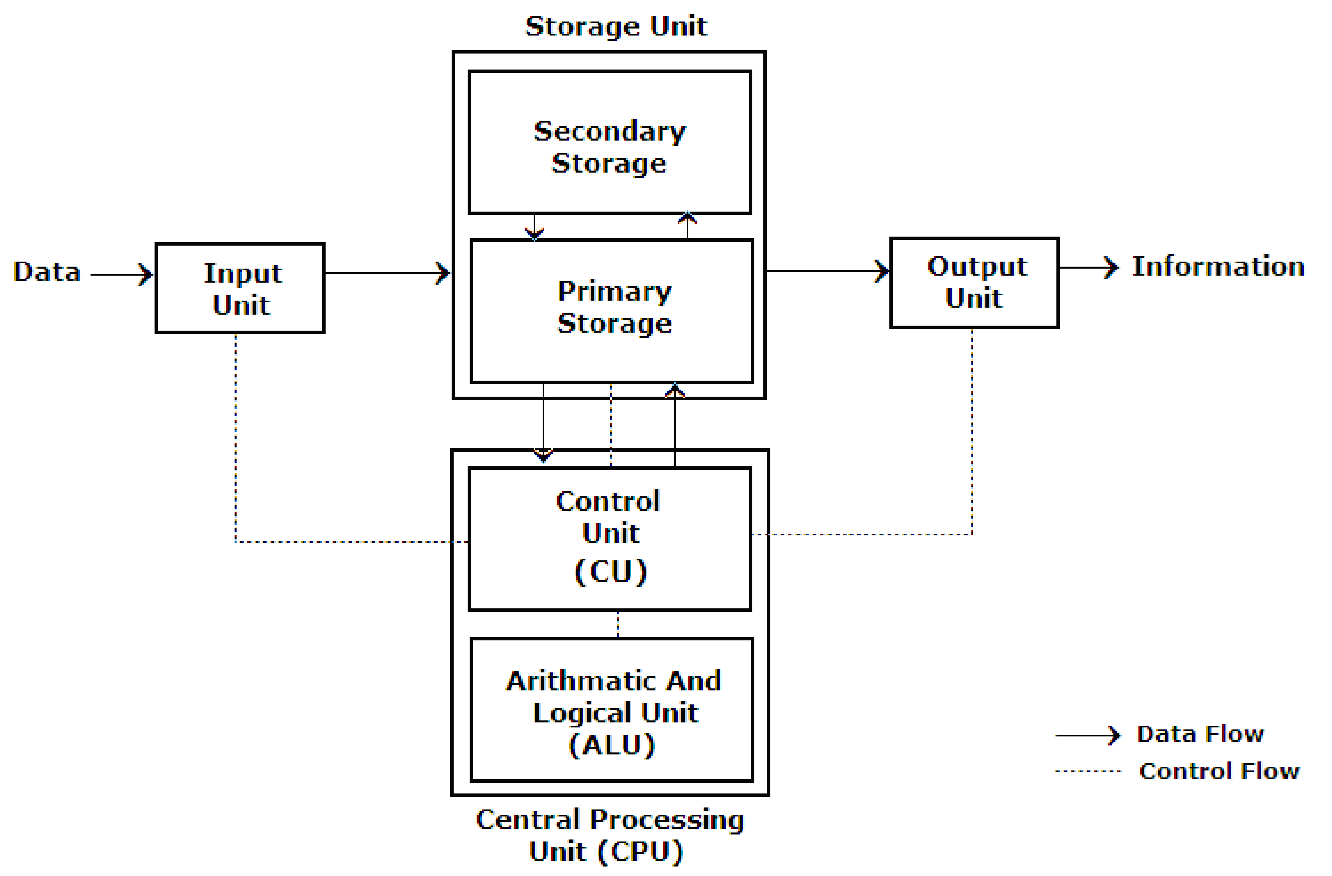
C++ was designed to organize the raw power of C using OOP, but maintain the speed of C and be able to run on many different types of computers. C++ is most often used in simulations, such as games. C++ provides an elegant way to track and manipulate hundreds of instances of people in elevators, or armies filled with different types of soldiers.

Other Languages

Besides these, now we have better languages such as Java, Perl, Ruby and Python.

1.3. Block Diagram of a Computer

A computer can process data, pictures, sound and graphics. They can solve highly complicated problems quickly and accurately.



Input Unit:

Computers need to receive data and instruction in order to solve any problem. Therefore we need to input the data and instructions into the computers. The input unit consists of one or more input devices. Keyboard is the one of the most commonly used input device. Other commonly used input devices are the mouse, floppy disk drive, magnetic tape, etc. All the input devices perform the following functions:

- Accept the data and instructions from the outside world.
- Convert it to a form that the computer can understand.
- Supply the converted data to the computer system for further processing.

Storage Unit:

The storage unit of the computer holds data and instructions that are entered through the input unit, before they are processed. It preserves the intermediate and final results before these are sent to the output devices. It also saves the data for the later use. The various storage devices of a computer system are divided into two categories.

1. **Primary Storage:** Stores and provides very fast. This memory is generally used to hold the program being currently executed in the computer, the data being received from the input unit, the intermediate and final results of the program. The primary memory is temporary in nature. The data is lost, when the computer is switched off. In order to store the data permanently, the data has to be transferred to the secondary memory. The cost of the primary storage is more compared to the secondary storage. Therefore most computers have limited primary storage capacity.
2. **Secondary Storage:** Secondary storage is used like an archive. It stores several programs, documents, data bases etc. The programs that you run on the computer are first transferred to the primary memory before it is actually run. Whenever the results are saved, again they get stored in the secondary memory. The secondary memory is slower and cheaper than the primary memory. Some of the commonly used secondary memory devices are Hard disk, CD, etc.

Output Unit:

The output unit of a computer provides the information and results of a computation to outside world. Printers, Visual Display Unit (VDU) are the commonly used output devices. Other commonly used output devices are floppy disk drive, hard disk drive, and magnetic tape drive.

Arithmetic and Logical Unit:

All calculations are performed in the Arithmetic Logic Unit (ALU) of the computer. It also does comparison and takes decision. The ALU can perform basic operations such as addition, subtraction, multiplication, division, etc and does logic operations viz, >, <, =, etc. Whenever calculations are required, the control unit transfers the data from storage unit to ALU once the computations are done, the results are transferred to the storage unit by the control unit and then it is send to the output unit for displaying results.

Control Unit:

It controls all other units in the computer. The control unit instructs the input unit, where to store the data after receiving it from the user. It controls the flow of data and instructions from the storage unit to ALU. It also controls the flow of results from the ALU to the storage unit. The control unit is generally referred as the central nervous system of the computer that control and synchronizes its working.

Central Processing Unit:

The Control Unit and ALU of the computer are together known as the Central Processing Unit (CPU). The CPU is like brain performs the following functions:

- It performs all calculations.
- It takes all decisions.
- It controls all units of the computer.

Other internal Units

Registers: It is a special temporary storage location within the CPU. Registers quickly, accept, store and transfer data and instructions that are being used immediately (main memory hold data that will be used shortly, secondary storage holds data that will be used later). To execute an instruction, the control unit of the CPU retrieves it from main memory and places it onto a register. The typical operations that take place in the processing of instruction are part of the instruction cycle or execution cycle. The instruction cycle refers to the retrieval of the instruction from main memory and its sub sequence at decoding. The process of alerting the circuits in CPU to perform the specified operation. The time it takes to go through the instruction cycle is referred to as instruction time.

Bus: The term Bus refers to an electrical pathway through which bits are transmitted between the various computer components. Depending on the design of the system, several types of buses may be present. The most important one is the data bus, which carries the data through out the central processing unit. The wider the data bus, the more data it can carry at one time and thus the greater the processing speed of the computer. Ex: Intel 8088 processor uses a data bus of 8 bits wide. Some super computers contain buses that are 128 bits wide.

Random Access Memory (RAM): The main memory of the computer is called as RAM. The name derives from the fact that data can be stored in and retrieved at random, from anywhere in the electronic main memory chips in approximately the same amount of time, no matter where the data is. Main memory is in an electronic or volatile state. When the computer is off, main memory is empty, when it is on it is capable of receiving and holding a copy of the software instructions, and data necessary for processing.

Because the main memory is a volatile form of storage that depends on electric power can go off during processing, users save their work frequently on to non volatile secondary storage devices such as diskettes or hard disk.

The main memory is used for the following purposes:

1. Storage of the copy of the main software program that controls the general operation of the computer. This copy is loaded on to the main memory when the computer is turned on, and it stays there as long as the computer is on.
2. Temporary storage of a copy of application program instruction, to be received by CPU for interpretation and processing or execution.
3. Temporary storage of data that has been input from the key board, until instructions call for the data to be transferred in to CPU for processing.
4. Temporary storage of data, which is required for further processing or transferred as output to output devices such as screen, a printer, a disk storage device.

1.4. Generations of Computer

The history of computer development is often referred to in reference to the different generations of computing devices. Each of the five generations of computers is characterized by a major technological development that fundamentally changed the way computers operate, resulting in increasingly smaller, cheaper, more powerful and more efficient and reliable computing devices. The five generations of computers and the technology developments that have led to the current devices that we use today. Our journey starts in 1940 with vacuum tube circuitry and goes to the present day -- and beyond -- with artificial intelligence.

First Generation (1940-1956) - Vacuum Tubes

The first computers used vacuum tubes for circuitry and magnetic drums for memory, and were often enormous, taking up entire rooms. They were very expensive to operate and in addition to using a great deal of electricity, generated a lot of heat, which was often the cause of malfunctions.

First generation computers relied on machine language, the lowest-level programming language understood by computers, to perform operations, and they could only solve one problem at a time. Input was based on punched cards and paper tape, and output was displayed on printouts.

The UNIVAC and ENIAC computers are examples of first-generation computing devices. The UNIVAC was the first commercial computer delivered to a business client, the U.S. Census Bureau in 1951.

Second Generation (1956-1963) - Transistors

Transistors replaced vacuum tubes and ushered in the second generation of computers. The transistor was invented in 1947 but did not see widespread use in computers until the late 1950s. The transistor was far superior to the vacuum tube, allowing computers to become smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation predecessors. Though the transistor still generated a great deal of heat that subjected the computer to damage, it was a vast improvement over the vacuum tube. Second-generation computers still relied on punched cards for input and printouts for output.

Second-generation computers moved from cryptic binary machine language to symbolic, or assembly, languages, which allowed programmers to specify instructions in words. High-level programming languages were also being developed at this time, such as early versions of COBOL and FORTRAN. These were also the first computers that stored their instructions in their memory, which moved from a magnetic drum to magnetic core technology.

The first computers of this generation were developed for the atomic energy industry.

Third Generation (1964-1971) - Integrated Circuits

The development of the integrated circuit was the hallmark of the third generation of computers. Transistors were miniaturized and placed on silicon chips, called semiconductors, which drastically increased the speed and efficiency of computers.

Instead of punched cards and printouts, users interacted with third generation computers through keyboards and monitors and interfaced with an operating system, which allowed the device to run many different applications at one time with a central program that monitored the memory. Computers for the first time became accessible to a mass audience because they were smaller and cheaper than their predecessors.

Fourth Generation (1971-Present) - Microprocessors

The microprocessor brought the fourth generation of computers, as thousands of integrated circuits were built onto a single silicon chip. What in the first generation filled an entire room could now fit in the palm of the hand. The Intel 4004 chip, developed in 1971, located all the components of the computer—from the central processing unit and memory to input/output controls—on a single chip.

In 1981 IBM introduced its first computer for the home user, and in 1984 Apple introduced the Macintosh. Microprocessors also moved out of the realm of desktop computers and into many areas of life as more and more everyday products began to use microprocessors.

As these small computers became more powerful, they could be linked together to form networks, which eventually led to the development of the Internet. Fourth generation computers also saw the development of GUIs, the mouse and hand held devices.

Fifth Generation (Present and Beyond) - Artificial Intelligence

Fifth generation computing devices, based on artificial intelligence, are still in development, though there are some applications, such as voice recognition, that are being used today. The use of parallel processing and superconductors is helping to make artificial intelligence a reality. Quantum computation and molecular and nanotechnology will radically change the face of computers in years to come. The goal of fifth-generation computing is to develop devices that respond to natural language input and are capable of learning and self-organization.

1.5. Types of Computer

Classify by principle of operation:

- Analog Computer
- Digital Computer
- Hybrid Computer

Classify by size:

- Micro Computer
- Mini Computer
- Mainframe Computer
- Super Computer

Classify by function:

- Servers
- Workstations
- Information Appliances
- Embedded Computers

Analog Computer

Analog Computer is a computing device that works on continuous range of values. The results given by the analog computers will only be approximate since they deal with quantities that vary continuously. It generally deals with physical variables such as voltage, pressure, temperature, speed, etc.

Digital Computer

On the other hand a digital computer operates on digital data such as numbers. It uses binary number system in which there are only two digits 0 and 1. Each one is called a bit.

The digital computer is designed using digital circuits in which there are two levels for an input or output signal. These two levels are known as logic 0 and logic 1. Digital Computers can give more accurate and faster results.

Digital computer is well suited for solving complex problems in engineering and technology. Hence digital computers have an increasing use in the field of design, research and data processing.

Based on the purpose, Digital computers can be further classified as,

General Purpose Computers

Special Purpose Computers

Special purpose computer is one that is built for a specific application. General purpose computers are used for any type of applications. They can store different programs and do the jobs as per the instructions specified on those programs. Most of the computers that we see today, are general purpose computers.

Hybrid Computer

A hybrid computer combines the desirable features of analog and digital computers. It is mostly used for automatic operations of complicated physical processes and machines. Now-a-days analog-to-digital and digital-to-analog converters are used for transforming the data into suitable form for either type of computation.

For example, in hospital’s ICU, analog devices might measure the patients temperature, blood pressure and other vital signs. These measurements which are in analog might then be converted into numbers and supplied to digital components in the system. These components are used to monitor the patient’s vital sign and send signals if any abnormal readings are detected. Hybrid computers are mainly used for specialized tasks.

Super Computers

Super computers are the best in terms of processing capacity and also the most expensive ones. These computers can process billions of instructions per second. Normally, they will be used for applications which require intensive numerical computations such as stock analysis, weather forecasting etc.

Other uses of supercomputers are scientific simulations, (animated) graphics, fluid dynamic calculations, nuclear energy research, electronic design, and analysis of geological data (e.g. in petrochemical prospecting). Perhaps the best known super computer manufacturer is Cray Research. Some of the "traditional" companies which produce super computers are Cray, IBM and Hewlett-Packard. Currently, China's Tianhe-2 is the fastest super computer in the world.

Mainframe Computers

Mainframe computers can also process data at very high speeds i.e., hundreds of million instructions per second and they are also quite expensive. Normally, they are used in banking, airlines and railways etc for their applications.

Mini Computers

Mini computers are lower to mainframe computers in terms of speed and storage capacity. They are also less expensive than mainframe computers. Some of the features of mainframes will not be available in mini computers. Hence, their performance also will be less than that of mainframes.

Micro Computers

The invention of microprocessor (single chip CPU) gave birth to the much cheaper micro computers. They are further classified into :

- Desktop Computers
- Laptop Computers
- Handheld Computers (PDAs, smart phones, tablets, etc)

More Info: [http://www.cs.cmu.edu/~fgandon/lecture/uk1999/computers_types/]

1.6. Types of Software

Various types of computer software are used to simplify the operations and applications of computer programs. Computer software enables the computer system to perform in accordance with the given tasks.

Basically, there are only 2 main types of Software.

- System Software
- Application Software

Besides these, software can also be categorized depending on the nature of use. There are 2 categories based on that. They are:

- Ready-made (off-the-shelf) Software
- Tailored (custom) Software

Also, we can categorize software under another broad heading as:

- Open Source Software
- Closed Source (Proprietary) Software

System Software

System software is the most commonly used variety types of software. System software offers a protective shield to all software applications. It also provides support to the physical components of computers. System software coordinates all external devices of computer system like printer, keyboard, displays etc.

System software sits directly on top of your computer's hardware components (also referred to as its bare metal). It includes the range of software you would install to your system that enables it to function. This includes the operating system, drivers for your hardware devices, linkers and debuggers. Systems software can also be used for managing computer resources. Systems software is designed to be used by the computer system itself, not human users.

Application Software

Application software is used for commercial purpose. The application software is widely used in educational, business and medical fields. Computer games are the most popular forms of application software. Industrial automation, databases, business software and medical software prove to be of great help in the respective fields. Educational software is widely used in educational institutes across the globe.

Applications software is designed to be used by end-users. Applications software, in essence, sits on top of system software, as it is unable to run without the operating system and other utilities. Applications software includes things like database programs, word processors and spreadsheets, e-mail applications, computer games, graphics programs and such. Generally, people will refer to applications software as software.

Ready-made (off-the-shelf) Software

These types of software are made and distributed by large enterprise companies. They are made with a general purpose requirement in mind. People using these kind of software may need to adjust in few features and usability.

Advantages:

- Lower up-front cost
- Contains many features, often more than you need
- Support is often included or can be added with a maintenance contract
- Upgrades may be provided for free or at reduced cost
- If it's software-as-a-service (SaaS) there is no hardware or software to install

Disadvantages:

- Slow to adapt or change to industry needs
- Your feature request may get ignored if it doesn't benefit the larger customer base
- May require you to change your process to fit the software
- Higher customization fees (proprietary software vendors often charge ridiculous hourly fees unless they provide an open API)

Tailored (custom) Software

These types of software are made with a specific customer in mind. They are given the opportunity to interact when in need to make any changes or adding any new features.

Advantages:

- You can start with the minimum necessary requirements and add on later
- Can be tailored to your exact business needs and processes
- Changes can be made quickly

Disadvantages:

- Very high initial cost
- All changes and feature requests will be billable
- May incur additional costs ramping up new developers

Open Source Software

These types of software provide have their source code available publicly. People can download, modify and re-distribute the modified versions of the software. They are maintained and contributed by thousands, if not millions, of people all over the world. They release very frequently and a number of features are added by different developers if they wish to do so. They release the bug fixes and other releases very quickly. They are generally free of cost as well, but not all of the open source software are free of cost.

Examples of some of the most popular open source software are Linux Kernel, Firefox Web Browser, Apache Web Server, LibreOffice Suite, etc.

Closed Source (Proprietary) **Software**

These types of software do not provide their source code to the general public. The developers work in a closed group and do not release the source code publicly. They are generally controlled by a particular enterprise company. These software generally charge money for license or subscription. But, there can be closed source software that are free of cost, they are called Freewares. The updates and bug fixes in these types of software are generally late as compared to open source software. People have to wait for the bug to be fixed by the company and release the updated version of the software.

Examples of some of the most popular closed source (proprietary) software are Microsoft Windows, Microsoft Office Suite, Internet Explorer, Skype, etc.

1.7. Types of Programming Languages

Fundamentally, languages can be broken down into two types: **imperative languages** in which you instruct the computer how to do a task, and **declarative languages** in which you tell the computer what to do. Declarative languages can further be broken down into **functional languages**, in which a program is constructed by composing functions, and l**ogic programming languages**, in which a program is constructed through a set of logical connections. Imperative languages read more like a list of steps for solving a problem, kind of like a recipe. Imperative languages include C, C++, and Java; functional languages include Haskell; logic programming languages include Prolog.

Imperative languages are sometimes broken into two subgroups: **procedural languages** like C, and **object-oriented languages** like C++ and Java. Object-oriented languages are a bit orthogonal to the groupings, though, as there are object-oriented functional languages (OCaml and Scala being examples).

You can also group **languages by typing: static and dynamic**. Statically-typed languages are ones in which typing is checked (and usually enforced) prior to running the program (typically during a compile phase); dynamically-typed languages defer type checking to runtime. C, C++, and Java are statically-typed languages; Python, Ruby, JavaScript, and Objective-C are dynamically-typed languages.

You can also **group languages by their typing discipline**: weak typing, which supports implicit type conversions, and strong typing, which prohibits implicit type conversions. The lines between the two are a bit blurry: according to some definitions, C is a weakly-typed languages, while others consider it to be strongly-typed. Typing discipline isn't really a useful way to group languages, anyway.

Different languages have different purposes, so it makes sense to talk about different kinds, or types, of languages. Some types are:

- Machine languages — interpreted directly in hardware
- Assembly languages — thin wrappers over a corresponding machine language
- High-level languages — anything machine-independent
- System languages — designed for writing low-level tasks, like memory and process management
- Scripting languages — generally extremely high-level and powerful
- Visual languages — non-text based

NOTE: These types are not mutually exclusive: Perl is both high-level and scripting; C is considered both high-level and system.

Machine Code

Most computers work by executing stored programs in a fetch-execute cycle. Machine code generally features :

- Registers to store values and intermediate results
- Very low-level machine instructions (add, sub, div, sqrt)
- Labels and conditional jumps to express control flow
- A lack of memory management support — programmers do that themselves
- Machine code is usually written in hex.

Assembly Language

An assembly language is basically just a simplistic encoding of machine code into something more readable. It does add labeled storage locations and jump targets and subroutine starting addresses, but not much more.

High-Level Languages

A high-level language gets away from all the constraints of a particular machine. HLLs have features such as:

- Names for almost everything: variables, types, subroutines, constants, modules
- Complex expressions (e.g. 2 * (y^5) >= 88 && sqrt(4.8) / 2 % 3 == 9)
- Control structures (conditionals, switches, loops)
- Composite types (arrays, structs)
- Type declarations
- Type checking
- Easy ways to manage global, local and heap storage
- Subroutines with their own private scope
- Abstract data types, modules, packages, classes
- Exceptions

System Languages

System programming languages differ from application programming languages in that they are more concerned with managing a computer system rather than solving general problems in health care, game playing, or finance. System languages deal with:

- Memory management
- Process management
- Data transfer
- Caches
- Device drivers
- Operating systems

Scripting Languages

Scripting languages are used for wiring together systems and applications at a very high level. They are almost always extremely expressive (they do a lot with very little code) and usually dynamic (the compiler does little, the run-time system does almost everything).

1.8. Traditional (Structured) Programming Concepts

Structured Programming is a method of planning programs that avoids the branching category of control structures. It is a technique for organizing and coding computer programs in which a hierarchy of modules is used, each having a single entry and a single exit point, and in which control is passed downward through the structure without unconditional branches to higher levels of the structure. Three types of control flow are used: sequential, test or selection, and iteration.

Traditional programming techniques focus on structures and separate functions that perform operations on those structures. Object-oriented programming treats the properties of a structure and possible operations on a structure as a single unit, called an object.

Thus the principal difference is the transitivity of the action: in traditional programming, the program performs operations on data, while in object-oriented programming, the program instructs objects to perform actions that in turn perform operations on data. OOP therefore introduces an additional level of abstraction over traditional programming techniques.

Principles of Structured Programming:

- Make flow of control as easily understood as possible. (Emphasis on use of Control Structures)
- Build your program from top-down. Decompose the problem into smaller and smaller pieces. (Top down programming)
- Avoid repeating the same code, can fix code in one place.

Top-down programming

Top-down programming refers to a style of programming where an application is constructed starting with a high-level description of what it is supposed to do, and breaking the specification down into simpler and simpler pieces, until a level has been reached that corresponds to the primitives of the programming language to be used.

Disadvantages of top-down programming

Top-down programming complicates testing. Noting executable exists until the very late in the development, so in order to test what has been done so far, one must write stubs .

Furthermore, top-down programming tends to generate modules that are very specific to the application that is being written, thus not very reusable.

But the main disadvantage of top-down programming is that all decisions made from the start of the project depend directly or indirectly on the high-level specification of the application. It is a well-known fact that this specification tends to change over time. When that happens, there is a great risk that large parts of the application need to be rewritten.

How does top-down programming work?

Top-down programming tends to generate modules that are based on functionality, usually in the form of functions or procedures. Typically, the high-level specification of the system states functionality. This high-level description is then refined to be a sequence or a loop of simpler functions or procedures, that are then themselves refined, etc.

In this style of programming, there is a great risk that implementation details of many data structures have to be shared between modules, and thus globally exposed. This in turn makes it tempting for other modules to use these implementation details, thereby creating unwanted dependencies between different parts of the application.

Bottom-up programming

Bottom-up programming refers to a style of programming where an application is constructed starting with existing primitives of the programming language, and constructing gradually more and more complicated features, until the all of the application has been written.

Advantages of bottom-up programming

Testing is simplified since no stubs are needed. While it might be necessary to write test functions, these are simpler to write than stubs, and sometimes not necessary at all, in particular if one uses an interactive programming environment such as Common Lisp or GDB.

Pieces of programs written bottom-up tend to be more general, and thus more reusable, than pieces of programs written top-down. In fact, one can argue that the purpose bottom-up programming is to create an application-specific language . Such a language is suitable for implementing an entire class of applications, not only the one that is to be written. This fact greatly simplifies maintenance, in particular adding new features to the application. It also makes it possible to delay the final decision concerning the exact functionality of the application. Being able to delay this decision makes it less likely that the client has changed his or her mind between the establishment of the specifications of the application and its implementation.

How does bottom-up programming work?

In a language such as C or Java, bottom-up programming takes the form of constructing abstract data types from primitives of the language or from existing abstract data types.

In Common Lisp, in addition to constructing abstract data types, it is common to build functions bottom-up from simpler functions, and to use macros to construct new special forms from simpler ones.