Introduction to Computers and Programming for Beginners

By Emmanuel Paige

Computers are machines that compute complex math problems and "one definition of a modern computer is that it is a system: an arrangement of hardware and software in hierarchical layers" (Ceruzzi). At first computers were large and cumbersome and dominated entire rooms in buildings, but currently they have been reduced to microstructures that can fit on the edge of a coin (Sakamoto). These machines have been integrated into many common devices we encounter in our daily lives. We see them all around us in the form of personal computers (PC), calculators, TVs, cell phones, ATMs, video game consoles, GPS devices, in cars, in appliances, in vending machines, and just about everywhere, as they are ubiquitous in the twenty-first century, and we rely on them to make our lives easier. These computing machines require technicians to maintain and repair the electronic hardware and programmers to input instructions in the form of software to make it function, and together they keep the system operational and computing information for the end user.

Computer programming is the process of inputting instructions in the form of source code that can be compiled into an executable application and perform calculations and process data. This is approached with the idea of creating software that will complete a specific task for computational tasks to solve problems, do calculations, and produce a desired effect and outcome by taking raw data, processing it, and then converting it to useful output. The information is processed directly by the central processing unit in machine code in the form of native instructions built into CPU and calculated at high speed in gigahertz which are billions of cycles per second. The executable programs are composed of code written in various languages that form source code with algorithms inside of functions and methods that return values according the needs of the programmer and end user. The field of programming often requires a diverse knowledge across several different subject including programming languages, logic, math, computer information systems, and algorithms. It is a high-tech field and competition can be fierce, although anyone can learn to program and gain skills necessary to create executable applications.

Although computer programmers are not necessarily technicians who repair and maintain hardware, the concepts are known to them because a fundamental understanding of how computers work is required to write software. Computer information systems (CIS) consist of a series of hardware devices combined into a functioning unit that utilizes electronic components in a system that is controlled by software that functions as an operating system to control all the integral parts and devices of the computer. These are the sum of components that work together in a personal computer (PC).

The hardware consists of all the internal components such as motherboard, central processing unit (CPU), memory (RAM), hard drive disks, solid state drives, internal network cards with RJ45 jack, WiFi cards, Bluetooth, CD/DVD disk drives, graphics card or graphics processing unit (GPU), and external peripheral devices such as keyboards, monitors, mice, tablets, scanners, touchscreens, printers, and any other device that can be plugged into the CIS to be used by the computer for input or output. The easiest way to plug peripheral devices into the CIS is through the USB port, although there are other ports such as IEEE 1394 Firewire, RJ45 network cable, WiFi, VGA, HDMI, DMI, and older COM and LPT ports.

The central processing unit (CPU)—the current CPUs in most consumer devices today are based on the x86 architecture instruction set. This is a CPU system developed by Intel that began with the 8086, a 16-bit processor, and then was followed by the 8088, and finally a series of processors that went from 80186, 80286, 80386, 80486 which created the x86 nomenclature. The Intel Pentium series of processor came next, and then the i series with i3, i5, i7, and i9
respectively. Intel i-series CPUs boast 64 bit, multi-cores up to 28 and 5GHz clock speed. There is competitor in the CPU market, American Micro Devices (AMD), and they also create CPUs which began around 1981 as a result of an agreement with IBM and Intel to create their own processors to fill a demand gap in the market (Justin). AMD microchips are high quality and they create CPUs and GPUs that are competitive in the market today with equivalent processing power and features to Intel’s products. The chips manufactured by AMD began in 1969 with embedded chips designated AM9080 And AM2900, moving up to AM29000 32-Bit RISC Processors, and then the licensed Intel clones AM286, AM386, AM486 And AMD 5×86. The true x86 AMD CPU began with the K-series: K5, K6, K6-II, K6-III, K7, K8, K10. At this point CPUs were 64 bit, multi-core, and had multi-gigahertz frequencies in stages until the AMD CPU Ryzen in 2016 which boasted 4-8 cores and 3.6 GHz clock speed. Now, in 2019, there are computers with CPU core counts upwards of 32 and with speeds reaching 5-6GHz. The race is on between the CPU giants, and it is getting crazy (Hruska).

Next is the motherboard, where all the electronic components are placed together to make a cohesive unit and is a microcosm of a small city with roads and tunnels and passageways for the electronic information to travel. Is it any wonder that the information travels on a hypothetical transporter known as the “bus” inside the system? Also, storage in the form of memory such as random-access memory (RAM), static memory, dynamic memory, and read only memory (ROM), which can be in the form of microchips or magnetic retrieval storage systems, like hard and floppy disks, compact disks, and magnetic tape. In their most basic form disks are circular shaped memory media that record data in magnetic form in sequence to be retrieved by the CIS on demand. The term disk is becoming obsolete as newer devices are using solid state drives (SSD) almost exclusively, and these are in the form of microchips and not a disk at all. Storage has taken many forms, including punched paper tabs, vacuum tubes, and even quartz crystal as a storage medium (Crisostomo). The future may hold memory techniques that we cannot even imagine at present.

Memory comes in many different shapes and sizes, and began as integrated chips fixed to the motherboard. Over time they required upgrades, so slots and sockets were created to make removing and installing RAM cards easier. The basic design of memory modules are on cards with pins which are inserted into slots. The types of RAM are DRMA, SDRAM.

Also, there is an output device, usually a monitor for viewing the information to be programmed or returned from the application. In the beginning output was in the form of light bulbs, diodes, and printed paper, although monitors were soon developed to view the output. Early monitors were cathode ray tube (CRT) devices in monochrome, a single color such as green or orange, that were simple and did not display multiple colors and were not very useful for vivid display. Input devices have always been necessary to communicate with the CIS in the form of keyboard, mouse, stylus, touch screen, and microphone (voice recognition is currently a popular technology).

Software is essentially a series of instructions that are a set of steps to be completed in order to tell the CPU and other devices what tasks are to be done. It is like using a recipe for cooking food or baking a cake. You begin by reading the instructions, combine all the ingredients in sequence, and follow the procedures, step by step, until you have a finished, fully baked cake. Software currently comes in two forms: open source is code that is open to the public, free, and can be modified; closed source is closed to the public, usually requires a fee, is proprietary, and cannot be modified.

Some familiar software are word processors such as MS Word, Corel WordPerfect, Apache OpenOffice Writer, also database programs such as MySQL, MS Access, Apache OpenOffice Base, and also web browsers such as MS Internet Explorer, Mozilla Firefox, Google Chrome, Opera, Apple Safari, and finally there are many video games, but World of Warcraft, Call of Duty, and The Legend of Zelda are a few popular examples.

Operating systems and basic input and output (BIOS) functions are software that is essential to make the computer operate (Ceruzzi). The BIOS is the first system to start when a computer powers up, or "boots", and it is a series of instructions and routines that connect all the devices in the CIS. This startup system resides in a memory chip (EEPROM) that is attached to the motherboard and is found on all electronic devices that have a microcomputer and require a boot sequence and software to function. After the BIOS is loaded, a true operating system, such as a disk operating system (DOS) is loaded to allow the user to interact with the computer and input and retrieve information.
Today, there are numerous advanced operating systems such as Microsoft Windows, Apple OS X, Unix, and Linux, to name the most popular in use at present. Although DOS was not the first true operating system, it is probably the software that singlehandedly changed how computers operating systems work. Invented by Xerox, pioneered by Microsoft with MSDOS, and ubiquitous across the first generation of IBM desktops personal computers (PC), DOS became the foundation upon which all operating systems now function.

Windows began as a shell over MSDOS and Windows 3.11 was a milestone operating system, changing the way humans interacted with PCs. Windows 95 became a true kernel operating system and no longer used DOS with a shell. Everything changed after Windows 95; although, the current version of Windows 10 still has the same basic kernel system intact since the 95 version.

Unix and Linux are virtually similar, except one is closed source and used by corporations and institutions for scientific and engineering purposes, and the other is open source and free to the public, used by many companies and private individuals alike. Unix was the operating system that set the standards early on in computer information systems and was created by AT&T. Linux was invented primarily by Linus Torvalds, a Finish-American software engineer, who created an alternative to Windows, based on Unix (Linus and Unix combine to create Linux), that was an open source kernel that directly competes with other paid, closed source, operating systems like Unix, OS X, and Windows. It is free to use on PCs and any other device that requires an operating system based on x86 chipset architecture. It is the primary operating system for Android phones, Chrome OS, and other experimental devices that require an operating system.

All software from operating systems to video games are created with a language by a programmer and compiled into an executable code for the end user to utilize as an application. Programming Languages come in two basic types: Interpreted languages (e.g. Assembler, BASIC, Python, Ruby, PHP, and JavaScript*), have an interpreter that can execute the code on the fly without being compiled into machine code first. The risk involved with these languages is that they will run with errors in the code, making it harder to debug. Also, these languages are generally slower than compiled executable programs. Compiled languages (e.g. Visual Basic, C, C++, C#, Fortran, Java, Lisp, and Pascal) are converted into machine code and execute much faster since they are at the basic machine level of instructions. They will not compile with errors, so debugging is easier. They have the disadvantage of being platform dependent, meaning that they can only run on the platforms on which they were compiled. High level languages are those like Visual Basic and C++ and Java that have easy to read instructions and data types names in human readable form. Programmers can use naming schemes and commands that make sense with long words and names to represent data. The down side is that these languages are hard for the computer to understand and they must be compiled down into machine code for execution. Low level languages such as Assembler have a low level of abstraction and can be converted directly to machine code without an interpreter or compiler. Another advantage is that they are fast and machine friendly. The downside is that they are hard to understand from the human perspective because they do not use large words and data type names that are user friendly. In assembler all the commands are in three letter pneumonic representation.

Compilers and integrated development environment (IDE) are the software that combines a set of tools like compiler, debugger, code editor, graphics editor, and other useful features into an application that helps programmers complete their task of creating software. Microsoft Studio is a popular example, and it can be used to program in Visual Basic, C, C++, C#, Fortran, Java, Lisp, and Pascal, and more, but it is a closed source system and can be quite expensive. NetBeans is another example, and it is open source and freeware, and can be used for most programming languages, and is particularly well suited for Java. There are other IDEs on the market, and it comes down to a matter of preference and choice for the programmer to decide which one to choose.

There are two basic program methods employed by programmers. The first is known as top down programming and is found in BASIC and some scripting languages like JavaScript and PHP where the code is executed from the top and executes down through the lines of code, starting at the first instruction, and following commands in sequence to the end. The second type is known as event-driven programming where the system executes an infinite loop and waits idle for an event to take place (Samek). The program is waiting for the user to do something, to start an event, and until then
the computer will wait patiently. Loops are possible in both forms of top down and event-driven programs; however, the latter is specifically designed to have an infinite loop and wait for an object or event to be triggered in order to respond. Top down programs terminate when it reaches an end statement or function, and event-driven programs end only when the user closes or terminates the program. To clarify, top down programs can be programmed to function like event-driven code, with loops and wait states, however, true top down programs will have an "end" statement and will complete at that point, whereas object driven programs end only upon the user terminating the application.

Object orientated programming (OOP) uses the abstract concept of objects and classes which are tangible things that can be used to trigger reactions in a program. The software starts at a main function and then waits in a loop for the user to select an object in the form of a button, textbox, widget, or any input interface like the keyboard, mouse, or touch screen, and then makes an appropriate response according to the logic involved in the routine surrounding the object. Instructions are the core routines preprogrammed into the CPU by the manufacturer (e.g. Intel or AMD) that software uses to perform procedures when at runtime. Commands are entered in the form of programs that can be “ran” or entered in at the command line and are executed in immediate mode, meaning that the IDE will provide instant reactions to the instructions on the fly. This is useful for doing calculations in real time and testing and debugging.

Information, in its purest form, is the primary concern for the programmer and the design and implementation of computing processes from algorithms and functions where information in the form of numbers and letters combine to form data are executed and produce output for the end user are the ultimate desired result. This information, or data, is input into the computer from storage media, nonvolatile memory (ROM), and peripheral devices and stored temporarily in the volatile memory (RAM) to be processed as a set of instructions. programming uses a language with syntax to process instructions that “crunch” information, or data, that is alphanumeric and consists of numbers and letters.

Information comes in the form of numbers with integers and decimals (floating-point), characters representing alphanumeric data, or strings of characters. The tiniest unit of information in computers is the bit. It is a single value represented by a 1 or a 0 and represents a binary switch with an on or off state. In its simplest form, information resides in memory on a computer as a state of on or off switches with an electrical charge keeping the state in the circuit. Once the switch is off, the electrical charge is gone, and the switch is now deactivated, or in a state of zero. The bit represents this value, like a light switch, it is either on or off.

The next most important unit of information in a computer system or program is a byte, which is a series of bits that come in different sizes (this is only basic information and can be different other architectures and CPU sets outside of the x86 infrastructure): A nibble is half a byte and is four bits in length. A byte is eight bits in length. A word is sixteen bits in length. A dword is a double word, and thus thirty-two bits in length. A qword is a quadruple word, and thus sixty-four bits in length. These are the basic units of information that form the building blocks upon which all programs are constructed and represented in memory in a computer.

The Binary number system is an important thing to consider because it is the system used to represent bits and bytes, at their most primitive level. Remember, there are eight bits in a byte. The way that binary number system count numbers larger than 0 or 1 is by placing a sequence of ones and zeros in a byte in an exponential sequence, with each sequentially adjacent place holder to the right of the number being the square root of the previous, resulting in a series of eight place holders like this 1^1:1^2:1^4:1^8:1^16:1^32:1^64:1^128:1^256. If you put a digit in any of these spots, it represents that number, for instance a 00001000 will represent the number 16. It looks like this for the number zero: 00000000. It looks like this for the number two-hundred and fiftysix: 11111110 . . . 100 = 001001100, and so on, up to 256 = 11111111. When the numbers exceed 256 at 8 bits, the byte width doubles, and we begin to use the byte values for a word (16 bits), dword (32 bits), or qword (64 bits). A sixteen-bit number in binary would appear thus: 1111111111111111 = 65535 in decimal. A 32-bit number would be double that in binary, and a 64 bit in binary double that again, reaching a potential value of 2^64 – 1 or 18,446,744,073,709,551,615 in decimal. This is an exceptionally large number for computers to deal with, and an unsigned integer this size requires an unsigned qword of 64 bit capacity.
There is another number system used in computer programming that is called hexadecimal and it is base sixteen which requires all ten numerals (0-9) and the first five letters of the alphabet (A-E) to represent numbers. The normal numbers zero through nine are to be at face value, however the number ten would be represented by A and sixteen would be represented by F. Any combination of these numbers will create a value as high as two-hundred and fifty-six which is represented by the letters FF, or 11111111. The numbers in hexadecimal, for example, will look something like this: 1A, 2B, 3C, AA, CD, EF, FF, and so on. It is used to represent machine code which is the basic language of the x86 CPU instruction set. All software is converted into machine code in order to control the system and is interpreted by the microprocessor, math coprocessor, and graphics processing unit. This is the native language of the computer.

Data types are data items with a collection of values, features, and attributes. "A data type is a type together with a collection of operations to manipulate the type" (Shaffer). Data types can represent integers, alphanumeric characters, arrays, structures, Booleans, objects, classes, lists, and abstract types by encapsulation. Data types are the primitive data forms that are implemented in programming to represent data in different ways, and show qualities and content possessed by each different type. A whole number is represented by an integer which is a basic data type known as int. Another abstract example: a light switch has two states, either on or off. This is a Boolean data type and has a value of true or false. The light switch, itself, is an object data type and would have member variables that represent these Boolean states of on or off—more specifically: true or false. A character or letter in the alphabet is represented by a character data type known as a char. An int and a char cannot be mixed logically and do not represent the same values in a program, although an integer can be represented as a char. The numerical value of a char is not the same as the integer value and will be return and ASCII code instead of a literal integer value. The basic data types can be combined in structures and classes with inherited data types that can combine through polymorphism to form extensive units of information in objects, which are the primary unit of data in object orientated programming (OOP). A program language requires these data types to be predefined and correctly associated with the proper values and properties in order to manipulate the data. If data types are assigned improper values a mismatched data type or incorrect data type error will be thrown by the compiler or debugger.

An integer is a number with a limited length of $2^{32} - 1$ and a qword length is any natural number from zero to $2^{63}$ or rounded off to nine quintillion. It is a number that is twenty digits in length. An unsigned long integer can hold this number, if it is a whole number and does not contain a sign, which is an unsigned number simply meaning that there is a minus sign in front of the number, hence, signed and unsigned, and it can only represent half of the available values in a long integer because half of the values are represented by negative numbers. Unsigned are long integers that contain no sign and can use all the places to contain the highest positive integer for the data type. Booleans (True/False) are logic operators that return a state of true or false, like binary logic, except that they only return these two states, true or false, and do not necessarily represent a numerical value, although zero and one can function as a binary return value. Characters are letters or numbers and are represented by the data type known as char. This is a one-byte unit that can hold a single character or number, but the value is always a char and not numeric for comparison procedures. Hence, a char representing a number cannot be compared to an integer containing a number value. Floating-point are basically decimals. If a number is not a whole number and contains a decimal point, then it is considered a floating-point integer, and this is the proper data type to use for these types of numbers. You will encounter errors if you assign values to the wrong data types with improper value that they are designed to incorporate.

Values in programming are represented by three basic forms: constants, which are literal values such as numbers or variables that have been assigned a constant value that cannot be changed; variables, which are abstract reallocated containers for values that have a label or name representing the constant value—they can be as primitive as a single letter or as complex as multiple words concatenated into CamelCase or Hungarian, which are the two most popular forms of notation within the naming convention nomenclature, in a phrase for ease and comprehension of the programmer (Mohanjo). Arrays are variables with multidimensional placeholders that can hold a collection of elements and has an index to represent the values on a grid system. Multidimensional arrays can be referenced by multiple indexes on an axis, and eventually will become a matrix in three dimensions.
Objects are abstract data types represented by classes. “An object is an instance of a class, that is, something that is created and takes up storage during the execution of a computer program” (Shaffer). Objects are generic data types that can represent a tangible item or widget like a button, textbox, slider, checkbox, radio button, cell, icon, and anything else that occurs on a computer user interface. If you can see it, touch it, click, move it, scroll it, or interact with it in any way, shape, or form, then it can be considered an object. Also, in programming, objects are anything that is acted upon or reverenced from in the data. An object such as a car will have many features that are member variables structured in the object. Think of an object to represent something tangible within a program.

Structures are datatypes that define attributes and characteristics of items, but lack polymorphism, and represent data in a more precise manner than a simple variable. Structures are like variables within variables and are represented by a dot notation. For example, a structure of a car may have members that represent the color, wheels, doors, horn, lights, and engine, and all these can be represented with the structure car.wheels, or car.doors, or car.horn, or car.lights. These can also be assigned different data types such as integer, string, Boolean, and such.

A class is an abstract representation of the object that has members and data structures that represent features related to the object. Classes can be encapsulated, and changed or inherited through polymorphism, which is a way to say that a base class or object can change and regenerate as a new and unique data type. A generic vehicle might be the object and the base class can be polymorphic when it is changed into different types of vehicles such as a car, truck, or bus. This is what give OOP and polymorphism such dynamic power in programming, because it is completely flexible to match any desired outcome or application.

Functions and methods are subroutines that are separate from the main program code branch and have their own internal logic and usually return a value. The difference between the two is that a method is a member of a class, and a function can stand alone on a global context. A function should only perform one routine and return a value for that operation to adhere to good programming principles. An example of a function might be to calculate the value of the area of a circle. You would send it a value of the radius of the circle and the function would calculate the formula of $3.14 \times \text{radius}^2$ and then return the answer as a value. The function would have a prototype with a name and parameters and then would be accessed with an argument, an input value, like this: calcArea(radius). The formula would be inside of the function, usually between an open and closed curly bracket. Here is a pseudocode representation of the above example:

```
float calcArea(float radius) { return 3.14 * radius^2 }
```

The pseudocode tells the compiler that this function will return a value of the calculated input radius. A method would be accessed similarly; however, it would use a dot notation accessor on the object to invoke the member function, or method. An object named circle might be accessed by calling on circle.calcArea(radius) and this would work much the same as the previously explained global function, only the formula is encapsulated inside the class or object.

Some famous and notable programmers are John G. Kemeny and Thomas E. Kurtz designed BASIC, Dennis Ritchie designed C and co-designed UNIX and B language, Ken Thompson designed UNIX and B language, Bjarne Stroustrup designed C++, Niklaus Wirth designed PASCAL, John Backus designed FORTRAN, Bill Gates designed Windows, Steve Wozniak designed Apple computers integrated with Integer BASIC, Linus Torvalds designed Linux, and Larry Page and Sergey Brin designed the Google search engine. The list goes on and on, and programming is a viable and worthwhile occupation for anyone interested in computer science.

In conclusion, programmers are responsible for making the applications that end users have come to know, enjoy, and love on their electronic devices. It started out in a primitive state with simple technology and made slow and steady advancements and was not always a glamorous undertaking and was the domain of the eccentric genius conducting wild hacking adventures, stealing code, and creating homemade computers in their garages before the advent of the ubiquitous PC, laptop, and cell phone came to be a household item. Today computers are fast and powerful, and technology is taking them to the next level of quantum computing where entangled quantum bits are being used to create lightning fast algorithms to generate random numbers and potentially will result in programmable quantum computer systems (Achieving Quantum Supremacy). Computer programmers come from all corners of the Earth and all
walks of life, but they are all intelligent and have problem solving skills and a desire to read endless lines of mind-boggling code to help them achieve their goal: an executable program without bugs, with an intuitive GUI and practical and desirable features ready for the end user. Elan Musk is a good example of a casual programmer who has achieved great status due to his desire to achieve his goals. He wrote his first program, Blaster, a space themed video game, at the age of 12 and sold it for $500 (Kim). He then went on to become a multi-billionaire with various ventures like PayPal, Tesla Motors, and SpaceX. Computer programming gave him an edge and it showed early on in his life that he was destined to be a smashing success. The future will hold great prospects and opportunities for programmers and computer scientists alike, and since technology is here to stay, this is a secure field of study and is a respectable and profitable occupation, especially for those with an entrepreneurial spirit, because the sky is, literally, the limit.

Works Cited


Samek, Miro. State Machines for Event-Driven Systems | Miro Samek 1 State Machines for Event-Driven Systems.

The computer stores a program while the program is running, as well as the data that the program is working with, in c. main memory. This is a volatile type of memory that is used only for temporary storage while a program is running. a. RAM. True or False Programmers must be careful not to make syntax errors when writing pseudocode programs. False.
is impractical for people to write programs in machine language. For this reason, assembly language was created in the early days of computing as an alternative to machine language. Instead of using binary numbers for instructions, assembly language uses short words that are known as mnemonics. For example, in assembly language, the mnemonic add typically means to add numbers, mul typically means to multiply numbers, and mov typically means to move a value to a location in memory. The main objective of this study guide is to provide an introduction to programming for people who have never worked with a computer before. The programming language used in the guide is C++. The main reason for this choice is that C++ is the main language used at Unisa for teaching programming. Firstly, you must have access to a computer, and you must sit down and use it as you work through this guide. There is no point whatsoever in reading through or trying to study from this guide without doing the activities practically. Programming is a practical skill (like learning to play tennis) and there is only one way to learn and that is to do it (as there is no way you can learn to play tennis without doing it!) A computer programmer is a computer scientist (a professional) skilled in using constructs of programming languages to develop executable and acceptable computer programs. A software developer is a programmer. Basic programming: nature and characteristics. BASIC is an acronym for beginners’ all-purpose symbolic instruction code. BASIC is the name of a high-level programming language developed at Dartmouth College, New Hampshire USA under the auspices of professors John G Kemeny and Thomas E. Kutz during their academic years 1963-64. The language was developed. To teach beginners the basic constructs of programming theory and as much, one of the easiest programming languages to learn.