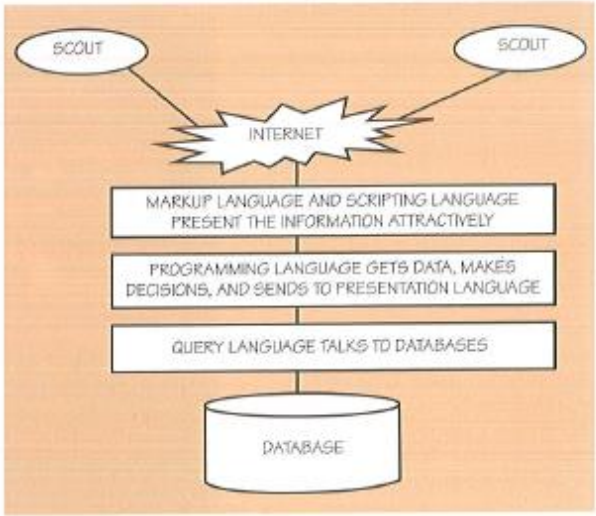
Web Languages

To understand the differences in Web languages, you must first understand how Web applications function. Once a request is received by a website, several layers must be navigated to present the information that was requested.



Databases

Databases are a critical part of a website. Adding a database to your website allows you to store, search, and organize large amounts of information. Think of it like a digital file cabinet containing many files. Leaving the files lying around creates a mess that is hard to organize. Putting those files into a well-organized system allows you to easily search and find what you're looking for. Databases help websites keep track of all the information and allow users to search and view that information.

Query Language

The query language is how you retrieve information from a database (files from the file cabinet) for presentation on-screen. To retrieve or alter information from the database, you use a query language like SQL (Structured Query Language).

Common commands for a query language include these:

SELECT allows you to search for the data you need.

INSERT allows you to add information to the database.

UPDATE allows you to make changes in the database.

DELETE allows you to remove information from your database.

Web Programming: Under the Hood

Once the query language gets the data, it hands the information off to the programming language for processing. This is where most of the programming happens. The programming language can decide what to do with the data: make decisions, search for specific things, or perform calculations. The programming language can also take a user's input, like a mouse click or keyboard entry, and make decisions. Think of this as asking and answering questions. The program asks: Do I have information to work with? What does the user want to do? Do I have the correct information? Do I need to change the information?

Internet programming languages offer many options. Choosing the language for a particular project often depends on the size of the project and the particular website features that will be needed.

The programming language is the powerhouse of Web development. It allows you to make changes to the data, process input from the computer screen, and do something with the data or input, if necessary.

SQL
Temperature
Example

```
Create proc [dbo].[GetTemperatureConversion]
@InputDegrees decimal(9,4)
as

select (@InputDegrees - 32.0) * (5.0/9.0) as DegreesCelcius
                ,case when @InputDegrees <=32 then 'Pack Long Underwear'
                      when @InputDegrees>100 then 'Remember to Hydrate'
                      else '' End as Instructions
GO
```

```
SQLQuery1.sql - S...ROOT\vrancce (54))*   SQLQuery2.sql - S...ROOT\vrancce (55))*  X
    DECLARE @return_value int

    EXEC    @return_value = [dbo].[GetTemperatureConversion]
            @InputDegrees = 100

    GO
```

Execution of the program

```
DECLARE     @return_value int

EXEC    @return_value = [dbo].[GetTemperatureConversion]
            @InputDegrees = 100

GO
```

Output of the program

Results | Messages

	DegreesCelcius	Instructions
1	37.7777400000	

Structured Query Language (SQL) is a special-purpose language designed for managing data in relational database management systems. SQL uses statements to execute database commands and to perform conditional operations and low-level functions. Unlike many of the other examples, SQL isn't designed to be a programming language and executes only when a user interacts with it.

Common Internet programming languages include these:

PHP—free, open-source. Easy to learn and used for small projects.

.NET—free and used for large-scale applications. Offers good performance, security, and stability. Limited to certain Web servers; Windows®-based.

JAVA—free and used for small and large-scale applications. Offers good performance, security, and stability. Able to run on different Web servers, such as Windows, Mac, and Linux.

Try this: Right-click on any Web page and select "View Source" or something similar, depending on the browser you are using, to view the source code of the Web page. You will see HTML and probably some JavaScript too.

Markup and Presentation: What You See in the Browser

When your programming logic is completed, next is the presentation layer or interface. This is what you see on-screen when you visit a website. Think of this step as presenting a nice picture. Typically, this final presentation is done with a combination of the languages, like adding style sheets, Hypertext Markup Language (HTML/HTML 5) for markup, and JavaScript for handling user input. These languages create what the user sees on the website.

HTML
(Uses JavaScript)

```
<!DOCTYPE html>

<html>
<head>
<title>Example Javascript Program for Boy Scout Merit Badge</title>
    <script src=' jsexample.js' type=' text/javascript'></script>
</head>

<body>

        <h1>Javascript Programming Exampe</h1>

            <h2>Enter Temperature (&deg;F):
                <input type="text" id="MyInputTemp"/>
                <input type="button" value="Go!" onclick="bePrepared() ;"/>
            </h2>
            <h3 id="myAnswer"></h3>

</body>
```

JavaScript
Temperature Example

```
function bePrepared() {
        var tempF = document.getElementById('MyInputTemp').value;
        var tempC = (5/9)*(tempF-32).toFixed(1);
        if (tempC < 7.2) {
                var myActionText="Take long-johns!";
        }
        else {
                if (tempC > 26.6) {
                        var myActionText = "Take sunscreen!";
                }
                else {
                        var myActionText = "Just have fun!";
                }
        }
        newText = "If the temperature is "+tempF+" &deg;F ("+tempC+" &deg;C): "+myActionText;
        document.getElementById('myAnswer').innerHTML = newText;
}
```

Javascript Programming Example

Enter Temperature (°F): 35 Go!

If the temperature is 35 °F (1.7 °C): Take long-johns!

JavaScript is a scripting language used to make dynamic websites. Free and easy to use, you already have it installed on your computer and can be programming in minutes. In this example, the HTML code formats the web page and makes it look nice. The second section of code supports that HTML and actually does the "heavy lifting." The JavaScript gets the temperature from the HTML input box, converts it to degrees Celsius, and displays a message depending on the current temperature.

Language Differences in a Nutshell

Programming language processes and organizes data and performs all the necessary decision making of your data to be handed off to the presentation interface. Examples: PHP, .NET, and Java

Markup languages describe how the data should be displayed on the screen, such as in bold, highlighted, or flashing green. Markup languages can be used simply to describe how the screen should be laid out, or they can describe all the coloring and other style elements of a Web page. Examples: HTML, CSS (Cascading Style Sheets), or XML (Extensible Markup Language)

Scripting languages, which can be embedded within HTML, are used to add functionality to a Web page, such as different menu styles or graphic displays, or to present dynamic (changing) information. Examples: JavaScript or jQuery

"HELLO WORLD!" EXAMPLES

JavaScript

```
<script type="text/javascript">
<!--function hello-World()
{
    alert('Hello World!');
}
// -->
</script>
```

SQL

```
Select * from myTable where text like "Hello World!"
```

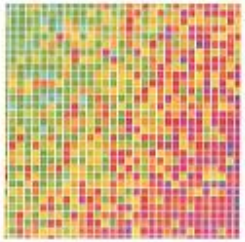
HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Hello World</title>
</head>
<body>
<h1>Hello World!</h1>
</body>
</html>
```

PHP

```
<? print("Hello World!"); ?>
```

Here is the same "Hello World" program in four different programming languages.



The word *pixel* is short for "picture element."

Animation and Computer Graphics

A picture on a computer screen or a cell phone is made up of individual tiny dots, or *pixels*. Pixels can be stored and moved together in one big block, as in an image that you might record with a digital camera. Pixels can also be updated individually by a program.

A program can read or write the value of any pixel in the image. This value is a number that has a range depending on how many colors a computer monitor or smartphone can display. The larger the number, the more colors are possible.

On a computer, colors are represented by numbers, and each "color number" is split into three color components: red, green, and blue. These three colors, abbreviated RGB, are combined the way a painter might mix several paints to get something in between. Using these three primary colors, displayed in various shades and strengths, any of 65,536 colors can be produced. For each color, 0 is the darkest, and 255 is the brightest when using eight bits to represent each color.

For example, when red, green, and blue are all dark—written as (0, 0, 0)—the color is equivalent to black; while RGB combined at each color's brightest—(255, 255, 255)—is equivalent to white. Red at its brightest with the other colors dark—(255, 0, 0)—produces red; but red displayed only "half bright"—(128, 0, 0)—gives dark red or maroon. Green at its brightest with the other colors dark—(0, 255, 0)—is a bright shade of lime, while green displayed "less than half bright"—(0, 100, 0)—produces dark green.

Following this pattern, what color do you think (0, 0, 128) represents? (Hint: It's a shade of dark blue with a nautical-sounding name.) What combination of red, green, and blue would produce violet? To check your answers, with your parent's permission, search online for an RGB color-code chart.

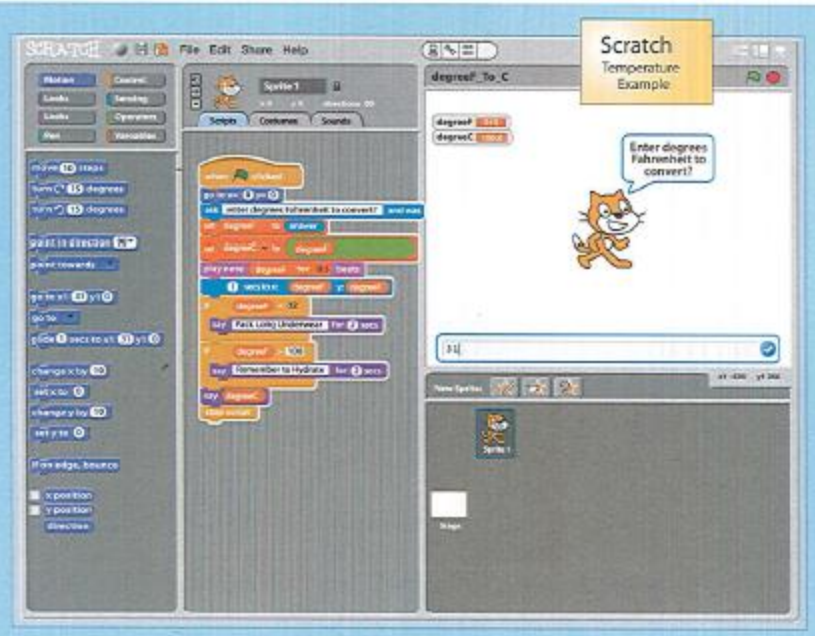
How are pixels used in animation? If you have several small images that are the same size (height and width), you can write a program to display the images one at a time at the same place on the screen, with a brief wait between each image in the series. This has the effect of flipping through the images. If the "flipping" is done fast enough, the sequence of individual images can appear to move on the screen. That is, there's "animation."

A *frame* can be an individual drawing in a comic strip, a single picture on a strip of film, or a single complete image for digital display. When a movie is "streamed" on a computer or tablet device, each frame is drawn (displayed) many times a second to create the illusion of movement. ("Streaming" lets you view data while it is being received rather than having to wait for it to download completely.)

Computer programs can also draw images from scratch when no picture has been taken with a camera. This is the basis for computer animation, which draws each frame of a movie or sequence using mathematics to define objects in the scene, lighting sources, direction of view, and other factors. When three-dimensional (3-D) images are drawn or rotated, the mathematics involved can be advanced, using a form of math called trigonometry. Special effects can also be added to an image frame that might be impossible to capture with a camera, such as explosions, extreme detail, or the merging of human actors with added special effects.

All of these 3-D image manipulations and special effects are processed by the computer program, pixel-by-pixel. The program must keep track of every pixel at all times during the animation sequence, which is a huge task since most images have millions of pixels.

Because animation is used in movies, video games, websites, smartphones, tablets, and other devices and applications, learning how to program animations is a good career opportunity that is also fun.



Scratch, which was developed by the Massachusetts Institute of Technology, is a great programming language for beginners. It allows you to quickly create simple animations and games by dragging blocks from the left screen, or the blocks palette, onto the center pane, or the scripts area. Scratch is easy to learn and fun to experiment with.

Entertainment

Programming is essential to many forms of modern entertainment, both at home and away. When you consider various high-tech amusements, you may be amazed by how much programming goes into the devices we use for fun.

At Home

At home, programmed devices have become the heart of the entertainment center. You might not recognize all of them.

Let's start with the television. Many modern TVs can connect to the Internet and stream videos or music using hardware or software designed for that purpose, without the need of a general-purpose computer. Some TVs can also display images and videos and play music stored on jump drives (small data-storage devices also known as flash drives or thumb drives). Such TVs have programming to handle these different forms of media.

Entertainment Programming Careers

In the entertainment industry, programming is vital, and it is big business.

Gaming. Video game programmers develop games for personal computers, tablets, smartphones, and game consoles. Gaming is a huge industry, with consumers spending billions of dollars each year on video games, hardware, and accessories.

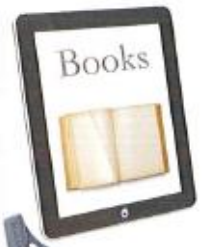
Movies and television. Media production may require complex programming for animations, computer-generated images, and motion capture (that is, recording the actions of human actors and using that information to animate digital characters). Media editing and producing optical discs (such as DVDs and Blu-ray™ discs) also may involve programming.

Besides connecting to the Internet, televisions may connect to many other programmed devices, including game consoles, disc players, and cable boxes.

Game consoles are special-purpose computers that run games and various kinds of media. The consoles include operating software and also execute specially developed game programs on removable media. Some consoles are programmed to support (work with) network-based multiplayer games.

Disc players such as DVD or Blu-ray™ players have programming to play DVD or Blu-ray videos. These devices are also often programmed to play audio or to connect with the Internet to stream video or audio.

Set-top boxes supplied by cable or satellite television companies are programmed to receive, decrypt, and stream audio and video transmitted by the cable company. These devices also include programmed user interfaces that allow users to configure and select various features. Some set-top boxes have digital video recording (DVR) functions that save video content for later playback. A large market for these devices has led to the development of operating systems specifically for devices that manage television content, such as the OpenTV operating system developed in the 1990s and now found in well over 100 million set-top boxes.



And finally, digital televisions can also connect to computers and double as large computer monitors.

Outdoors

Programmed devices for entertainment are not limited to the home. Many devices that people use outside also require serious programming. Programmed mobile computing devices allow people to communicate and play while on the go. Smartphones and tablets have all kinds of entertainment programming, from games to players for various types of media.

Programmed devices for entertainment include gadgets such as GPS units, music players, and e-books. Can you think of other programmed devices that people use for fun and adventure?

Science

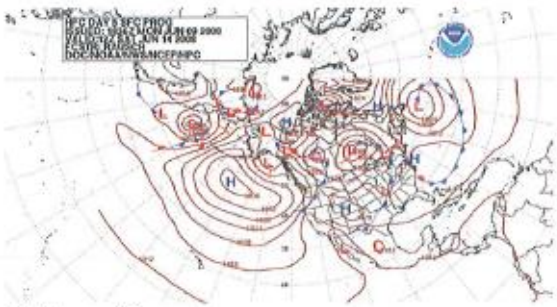
Scientists study the world around us to discover how it works. Modern science requires lots of computing power to collect data, analyze the data, and ultimately make sense of it all. Here are a few examples of programming in science. Many more exist.

Chemical Instruments

Chemists use various computer-controlled devices to test chemicals. These instruments can help determine what is in the samples being tested and how the chemicals behave. Programmers create the programs that control the devices so they can run for many hours without the operator needing to load new samples. One kind of instrument that is often used in chemical labs is the gas chromatograph. It helps scientists determine what is in liquid samples and how much of each chemical is present, using heat and gas pressure to separate the parts of the sample.

Materials Testing

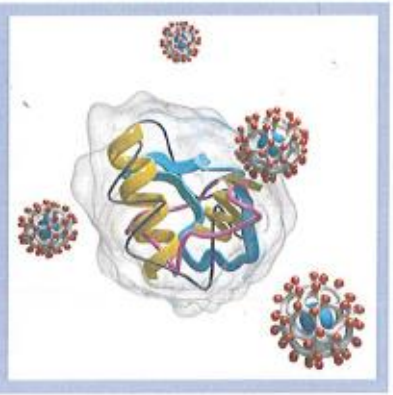
Laboratories that test plastic, metal, and ceramic parts use physical testing instruments to measure the strength and durability of the parts. These instruments use control programs to operate the clamps and motors that hold and manipulate the parts that are being tested. Programs are also used to analyze the data that is collected during the tests. These tests help materials scientists determine whether critical parts in products will be durable and safe for the people who buy the products.



Weather prediction

Meteorologists and programmers work together to create weather forecasts. Scientists use computer programs to represent the atmosphere as a grid of points with each point containing math equations that describe weather. Millions of points form simulations that compute how weather patterns will evolve. Meteorologists collect weather data from weather stations all over the world. The data is input into simulations of the atmosphere. Math equations and large volumes of data require powerful supercomputers to answer the question of whether or not it will rain today.

Researchers use computers to create new drugs that fight cancer. With too many possible combinations of chemicals to test each one, scientists use simulations to predict how molecules will interact with cells. By testing many different possibilities with computers, scientists move a step closer to a cure for cancer.



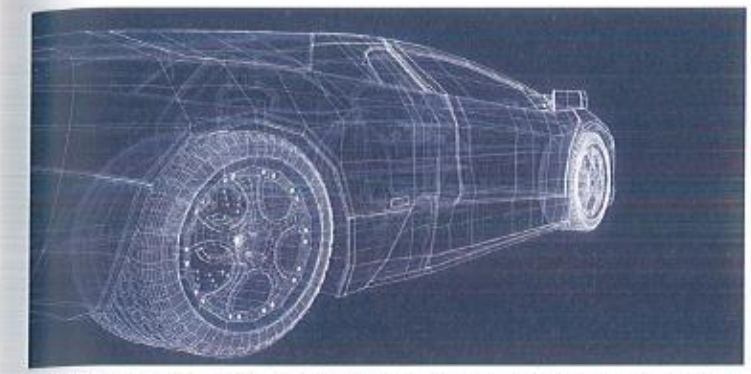
The Ruby programming language is used primarily for web application development. It is a simple, open source (free) object-oriented programming (OOP) language that is gaining popularity due to its flexibility and ease of use.

Engineering

Have you ever wondered how the space shuttle, rockets, or jet fighters were designed? Or how a bridge handles the stress of all those cars and trucks? Or how a building is designed to withstand an earthquake or hurricane?

Engineers apply science and mathematics to design and build dependable structures, systems, machines, devices, materials, and processes. The different fields of engineering include electrical, mechanical, civil, chemical, computer, and aerospace. All of these engineering disciplines require knowing how to program.

In fact, programs become more vital as engineering problems become more complicated. Engineers use programming to solve problems creatively and as inexpensively as possible. In the past, engineers had to rely on wind tunnels and scale models to test whether designs would work. Now, engineers use programs to create computer-generated models in place of physical mockups, which saves time and money and creates better designs. Whether for the tallest building in the world or the longest bridge, the plans can be designed and tested, all by computer.



To save time and money, CAD engineers can create and test new products in a virtual environment before production actually begins.

Computer-aided design (CAD) programs allow engineers to create models and schematics of their designs. The models can be placed into artificial environments where the engineer can virtually "walk" and move objects around. This solution avoids the need to build expensive physical models.

Programs have been created to check for flaws; measure the fit; and analyze the properties of things, such as stresses, temperatures, and movements. Customized programs have also been created for special purposes such as landing an astronaut on the moon, sending satellites into orbit, and producing heads-up displays for fighter pilots.

One thing common to all areas of engineering is problem solving. How big does a thing have to be? How strong? Engineers design things based on the knowledge they have. Information about something may not always be available, and this is where computers are especially helpful. Programs can quickly solve complex math problems that humans cannot figure out. Engineers use computers in problem solving so that they do not have to build something to find the answer. This saves money and time.

Objective C
Temperature
Example

```
#import "campout.h"
@implementation campout

//..general function to get string input from keyboard..
NSString *getStringInput(NSString *prompt)
{
        printf("%s", [prompt cStringUsingEncoding:NSUTF8StringEncoding]);
        NSFileHandle *input = [NSFileHandle fileHandleWithStandardInput];
        NSData *inputData = [NSData dataWithData:[input readDataT      oEndOfFile]];
        return [[NSString alloc] initWithData:inputData encoding:NSUTF8StringEncoding];
}

//..general function to get number value from keyboard. Gets string first, then
// converts it to a number...
int getNumberInput(NSString *prompt)
{
        NSString *inputString = getStringInput(prompt);
        return [inputString integerValue];
}
void howToPack()
{
        NSString *continueYN =   @"y" ;
        int nDegreeF =   0;
        int nDegreeC =   0;

        while ([continueYN isEqualT   oString:@"y" ])
        {
                nDgreeF = getNumberInput(@   "What temperature in degrees F:");
                nDegreeC = (nDegreeF -   32) * 5/9;
                printf("The temperature is %i degrees C\n",     nDgreeC);

                if (nDegreeC <=   0)
                {
                        printf("Pack an extra sweater! \n"   );
                }
                if(nDegreeF >   100)
                {
                        printf("Remeber to hydrate! \n"   );
                }
                continueYN = getStringInput(@   "Continue with another temperature? (y/n):"   );
        }
}
```

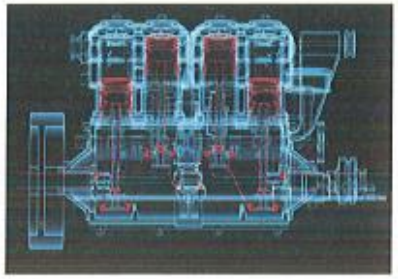
Objective-C is a compiled language that is derived from C and is purely object-oriented. This is a useful language to learn because the iPod, iPad, and other Apple computers use this as their native language for programs and apps.

Engineering Programming

As an example, consider the design and building of an aircraft. Many different engineering talents are needed. Aerospace engineers work on the design and construction of the aircraft; they must understand the science of flying. Electrical engineers design the electrical systems that operate everything from the wheels to the lights. Computer engineers create the programs that control many of the devices on a plane, including navigation systems, flight controllers (autopilot), and temperature controls. Computer engineers also help visualize the designs of new aircraft using CAD tools.

To solve engineering problems such as aircraft design, general-purpose programming languages may be used, including Fortran, C, C++, BASIC, and Pascal. Specialized software helps with the complex mathematics involved. Engineering and scientific programming also uses object-oriented programming, parallel programming, and various modern languages such as Java and Ada.

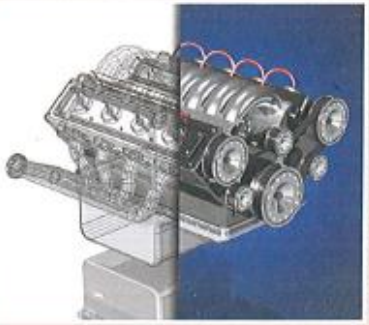
Whatever engineering discipline you might choose to pursue as a career, all fields of engineering require knowledge and ability in programming. Programming can open the doors to many career opportunities in engineering, from building race cars and fighter jets to ensuring that buildings won't fall in an earthquake or a tornado.



Automobiles and Traffic Control

Most cars today have dozens of computer controls for safety, performance, fuel economy, and driver convenience. These computers exchange information and work together—a built-in communications network right in the car. Multiple networks may be dedicated to different parts of the car: one network for the engine and transmission; another for the cabin controls and dashboard. Then a separate computer is programmed as a dedicated communications controller between the networks. Programs even fine-tune the engine for better performance and fuel economy. All these computers work together to operate different features of the car such as the engine, transmission, dashboard display—even the windows. Automobiles today require *lots* of programming!

An example of programming for safety is the antilock braking system (ABS). Drivers used to be taught to pump the brakes to maintain control of a skidding car, especially in snow or ice. With ABS, the programming controls the on/off action of a brake much faster than a person could. This allows for better control in a skid. Even on a dry road, a driver making a very quick stop might hear or feel the rapid operation of the ABS.



The processors in a typical modern car require an estimated 50 million lines of programming. Luxury cars with up to 100 processors could have 100 million lines of programming to handle features such as automatic parking, hybrid drivetrains, tire air pressure, collision avoidance, etc. Some cars have as much programming as a modern fighter jet or commercial aircraft.

For security, a car may have a program that communicates with the remote key control. A driver can remotely lock and unlock the car, open windows and a sliding door, and even set off the car alarm.

Additionally, there is much programming for the dashboard, or instrument panel. The displays may be digital, and a driver information screen may present messages. Also, the radio cluster may have GPS, maps, and satellite radio, all operated by a touch screen. The doors may have a separate program to control window motion. Some cars have automatic one-touch operation of windows.

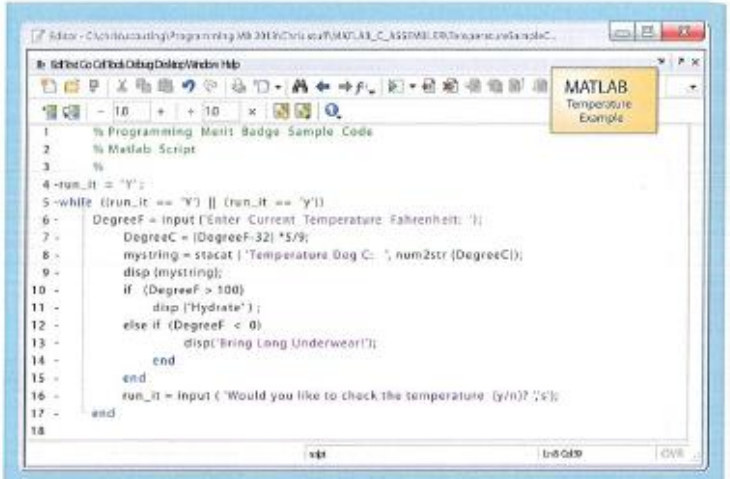
None of these features for comfort, convenience, or security would be possible without a computer program that someone has written. The program senses the environment, makes decisions on that information as well as user commands, and acts accordingly. All these actions must be done safely, quickly, reliably, and repeatedly, as expected by car owners.

Traffic Control

Traffic control on roadways is necessary in order to have many cars operate safely together. Red-yellow-green signal lights are a national standard in the United States and in many other countries. These traffic-control devices are programmed to safely control the sharing of intersections by cars traveling on crossing roadways.

A traffic-light program has multiple inputs. Its own timer program controls the progress of the program. Also, there may be a communications connection to an external processor. The program can be developed and tested on that external processor, then downloaded into the traffic-controller processor and run. The output would be the signals to control circuits that activate and deactivate the traffic lights. These control programs can also be used to coordinate multiple signals at multiple intersections to improve the flow of traffic.

If a program is designed well, it's easy to add more features such as left-turn control from one or both streets. Another possible feature is for blinking lights overnight, when traffic is sparse. The timing of all the traffic lights could be adjusted based on other inputs such as sensors in the road. A roadbed sensor would detect a car's approach and adjust the traffic lights accordingly.



Command Window

```
>> TemperatureSampleCode
Enter Current Temperature Fahrenheit: 72
Temperature Deg C:22.2222
Would you like to check the temperature (y/n) ? y
Enter Current Temperature Fahrenheit: 101
Temperature Deg C:39.3333
Hydrate
Would you like to check the temperature (y/n)? Y
Enter Current Temperatue Fahrenheit: -20
Temperature Deg C:-28.8889
Bring Long Underwear!
Would you like to check the temperature (y/n)? n
fx >>
```

Matlab is a scripting language with lots of powerful numeric analysis functions. Matlab is widely used in the science, engineering, and mathematics fields.

Transportation Safety

Programming can be simple or complex, depending on the purpose. A program can be designed to adjust its behavior based on environmental input from sensors. Programs for safety-critical machines need to be tested many times and in different situations to be sure they operate properly, reliably, and safely.

Programming is not limited to automotive transportation. Other means of transportation such as airplanes, trains, ships, and mass transit require just as much or even more programming. As you can see, programming is essential to all areas of transportation.

Computer and Information Security

Hackers are people who try to bypass the normal design of a program or stop a computer from working altogether. They often start out as programmers and experiment with the boundaries of what a computer can do, but wind up working for criminals. These hackers use the Internet as their highway into many computer systems.

Computer security is a growing and rapidly changing area of computer programming. To protect computer networks and the businesses that depend on them, programming is done to encrypt (encode) data; check the user's credentials; collect and analyze logs of data activity; develop and design new and better biometric sensors (to read a user's fingerprint or voice patterns, for example); and design fail-safe devices for automotive and aeronautical systems, trains, law enforcement, and public safety.

A constant battle goes on in cyberspace between the hacker and the defender. But the protector of the computer system often (not always) gains an advantage because hackers can go only where the wires let them. If all the routes are known and protected using best practices, then the computer and network will be less vulnerable to hacker threats.

The "CIA" of Computer Security

One rule among programmers is that security should be designed into a program from the beginning. That is, try to think about how someone might intentionally cause the program to not work as expected, and write the program to handle all situations.

When designing for security, programmers keep three principles in mind: *confidentiality*, *integrity*, and *availability*. These principles are sometimes referred to as the "CIA" of computer security.

Confidentiality means that only the people who are authorized to see information are able to see it. Confidentiality can be protected by requiring passwords to computer accounts, using encryption to hide the data from plain viewing, or keeping a laptop or thumb drive in a locked desk drawer.

Integrity means the data being handled by the program is not altered from how it was originally stored or transmitted. An example is a bank program that processes a transfer of $100 from one account to another. If the program did not check data integrity and a hacker got into the system, the hacker might change the amount and transfer $10,000 instead. Fortunately, there are ways to check if data has changed along its path.

Availability means the computer is up and running and able to process instructions. A common Internet attack that uses mobs of "zombie computers" (computers that have been secretly taken over by a hacker) is known as a distributed denial of service (DDOS). The "denial" happens when many computers are coordinated to request the same Web page from the same server at the same time. This overloads the Web server and can cause it to crash. While the server is offline or is rebooting, no one can use it. Some businesses process millions of dollars a day using their Internet connections, so being offline for a few hours can mean a huge loss of money.

Home Computer Security

These principles of computer security are used every day in business, but can also be used on your own laptop, your computer at home, and even a smartphone.

- If you have a home network, you can ensure its **confidentiality** by making sure it is running with strong Wi-Fi encryption on the wireless access point.
- You can help to ensure the **integrity** of a computer by running an update on the operating system to get the latest security patches. These are usually released by the software's seller for free to plug holes or weak points that might allow a hacker to get control of it.
- You can safeguard a computer's **availability** by running a backup of the disk and data files. A backup gives your data some protection from a hardware failure or a lost laptop or device. The data files can be loaded onto a replacement device. A spare battery or a power adapter are other ways to maintain the availability of a mobile computer.

The most advanced area of computer security programming is called reverse engineering. This involves examining a compiled module and trying to determine the code that was used to produce it. Most often, a programmer using reverse-engineering skills will analyze a virus that has infected a computer to figure out how to neutralize and remove the virus, and also determine what kind of problems it could cause.

Positive Identification

Another growing area of computer security involves user authentication. A concept called *two-factor identification* guarantees a person actually is the person he or she claims to be. This assurance is becoming increasingly important as people do more business online and do not actually meet each other face to face.

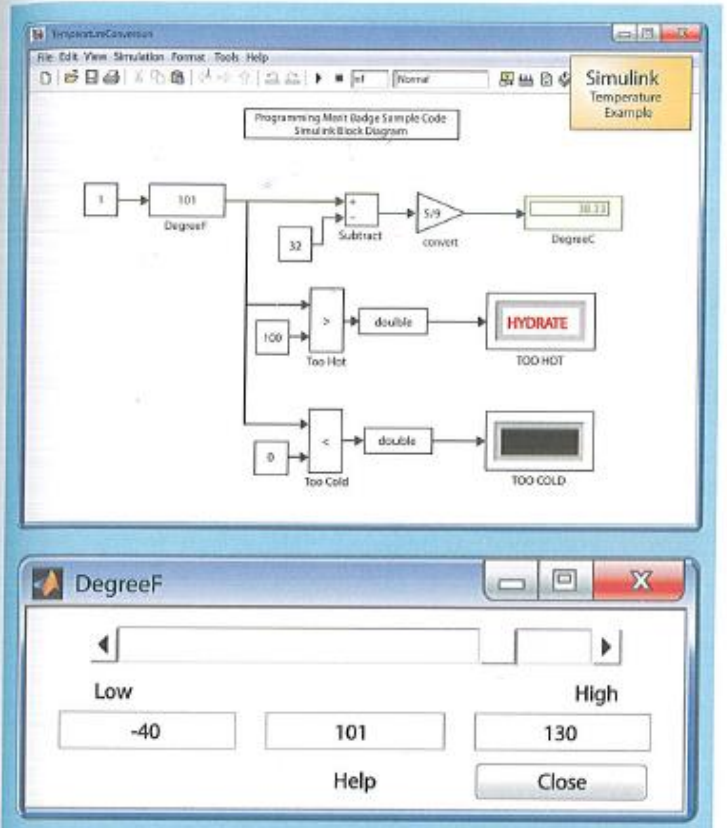
Two-factor identification means that any two things from this group can be used:

- Something you know (a secret, like a password)
- Something you have (a hardware token or a smart card)
- Something you are (a biometric characteristic like a fingerprint, an iris scan, or a facial scan)

If only one of the above is used, a hacker could copy or steal just that detail; the most common single-factor security system uses only passwords. But when a second factor is added, a hacker is far less likely to get both details. The most common form of two-factor identification is the smart card, which is in the device itself. A code to unlock the device, which you can reset, is the "something you know." This means that if you lose the card or the device, a thief cannot use it.

A fast-growing area of computer hacking is called *identity theft*. An identity thief uses stolen information to create a fake profile of someone and attempts to do illegal things with it, such as obtaining credit and buying merchandise in the name of the victim. An imposter might give a false identification to the police, creating a criminal record for the person whose identity was stolen. Or the imposter could use the victim's email contacts to send people phony, unwanted, and often offensive or malicious emails.

For these reasons, computer programming to help people safeguard their identity (and protect their money and reputations) will be a growing area in the future.



The Simulink programming language uses a block diagram. In this example, the program is launched from the main window by selecting the source file and is executed by using the "play" arrow. The user then selects a value by double-clicking on DegreeF to get the slider.

Training for a Programming Career

Along with excellent the ability to think logically and solve problems, a good programmer also needs patience and creativity. Complicated programs demand creative approaches and patient attention to detail. A programmer also needs good communication skills and the ability to work in teams. Equally important, however, is the ability to work alone for long periods of time. Programming can be a lonely and isolated type of work.

Programmers are constantly learning. In the fast-changing world of technology, they must keep up with the latest advances and master complex new languages that are only now being developed.

Educational Opportunities

Some programmers are self-taught. Programming languages can be learned through online courses, guidebooks, and home study. Writing code and working out the bugs on your own is a straightforward way of getting practical experience.

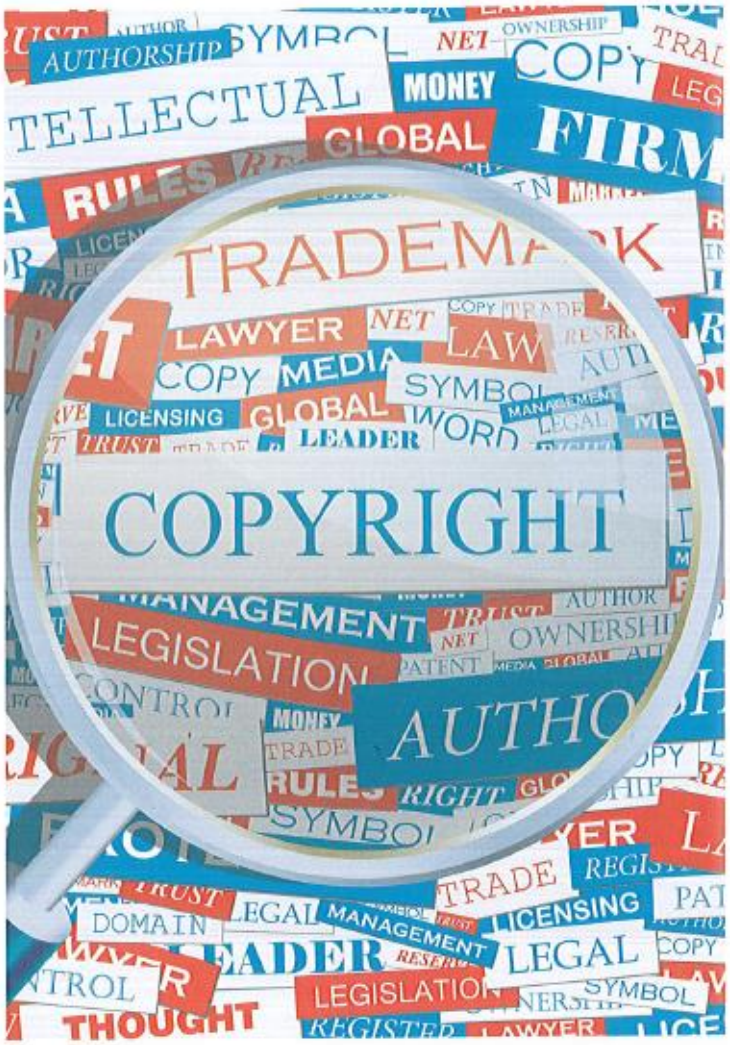
When you have chosen the one career opportunity in programming that most interests you, your merit badge counselor or school guidance counselor can help you identify the specific education, training, and experience your chosen field requires.

Careers in Programming

As you can see from all the different languages and industries covered in this pamphlet, programming is used EVERYWHERE. And that means programmers are used everywhere, too. It would be hard to find an industry today that doesn't have some need for programmers.

Programmers can be self-taught or have college degrees. Regardless, with programming you are never finished with learning—there is *always* something new to discover. This may explain why the biggest factor in getting programming work is experience, and the more you can get, the better off you will be.

While you can make programming your career focus, more and more today, programming is becoming just another tool you use to get something else done. Very often people choose a field of study that interests them (not programming specifically) and then learn how to use programming to support their line of work. So whether you become a hard-core programmer or just use programming as a tool to complete a task, programming is an awesome skill to have!



Intellectual Property

When you create a new product, like a bicycle, you have a physical object you can touch and hold. When you create a computer program, however, you can hold the physical media that stores the programming but you cannot hold the programming itself. With the bicycle, the value to people is in the physical object. But with computer programs, the value is in the programming, not the physical disc or other media on which the program is stored.

Intellectual property, or IP, refers to creations that cannot be touched or held, such as ideas, plans, and designs.

Intellectual Property Rights

Over time, different kinds of rights have been developed to protect items you cannot touch and hold. IP rights include copyright, patent, trademark, and trade secret. Different aspects of computer programs may be legally safeguarded by each of these forms of protection.

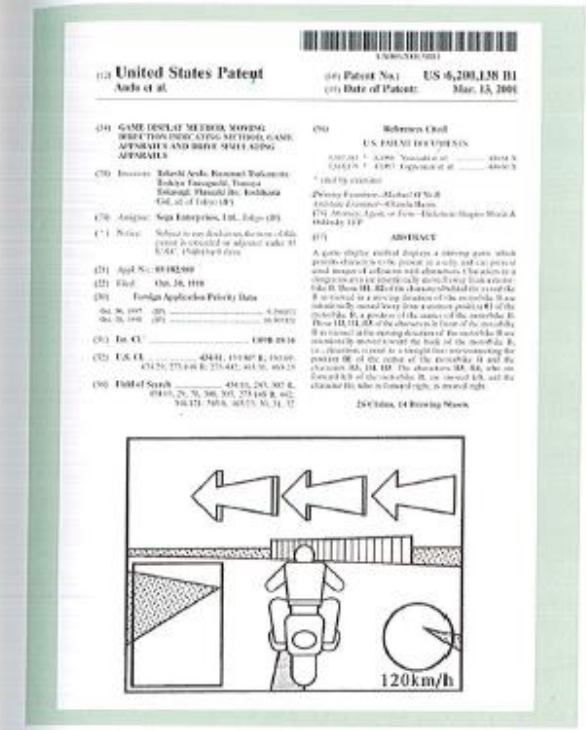
Copyright Protections

A copyright protects a particular expression of an idea that an author has created. The author gets the copyright to the work when it is created and stored in some tangible media. For example, in a game involving a spaceship flying around a screen shooting asteroids, the programmer would get copyrights to several distinctive and original aspects of the game once the game is created.

First, the programmer would get a copyright on the particular background screen showing space and obstacles. The copyright would be limited to the unique background screen created by the programmer, so other people would be free to create their own background screens with space and obstacles. If someone copied the particular background screen from the copyrighted game, the programmer could rely on the protections of copyright to stop that other person from using the copied background screen.

Also, the programmer would get a copyright on the actual set of instructions that causes the game to operate. Someone else could create programs to perform similar game concepts as long as the other person did not copy the exact set of instructions developed by the programmer.

The Digital Millennium Copyright Act of 1998 extended the protection of copyright holders by imposing penalties for attempts to violate copy protection. Copy protection, also known as content protection, aims to stop the illegal reproduction of computer software as well as movies, music, and other media.



Patents

A patent protects useful and innovative processes or methods, machines, manufactured items, or "compositions of matter" (things made of two or more substances or ingredients). Unlike copyrights that are automatic when the work is created, patents must be applied for. To seek patent protection, a programmer must file an application with the United States Patent and Trademark Office. A government patent examiner reviews the application to determine whether it is for something truly new, useful, and not obvious. The application process is often lengthy and typically requires the services of a patent attorney or a patent agent.

As an example, think about the asteroid shooting game mentioned earlier. If aspects of the game are innovative, the programmer may be able to get patent protection. The programmer could then use the patent to stop others from making, using, selling, importing, or offering for sale the programmer's patented invention.

Trademarks

A trademark protects a word, phrase, symbol, sound, or color—the "mark"—that identifies and distinguishes the source of a particular product or service. You receive trademark rights by using the mark in connection with the sale of a product or service. Those rights develop over time as customers begin to connect the mark with a product.

People often use a superscript TM (™) next to their mark to put others on notice that they are claiming trademark rights. If the owner of a mark registers it with the United States Patent and Trademark Office, the owner of the mark may use a ® to show that the mark has been registered.

Programmers often use trademarks to keep others from using a similar name for a competing product. For example, if the programmer named the asteroid shooting game "Astroblitz," he or she might want trademark protection to make sure someone else does not sell a competing asteroid shooting game with the same or a very similar name.

Trade Secrets

A trade secret protects commercially valuable information for which the owner has taken strong measures to maintain secrecy. This protection will exist as long as the information remains a secret.

In the example of the asteroid shooting game, the programmer could keep the source code a trade secret. To do so, the programmer would need to strictly control access to the source code. He or she would need to put everyone who has access to the code under an obligation to maintain the secrecy. This might be done by having everyone with access sign a contract promising not to reveal the secret.

Unlike a patent, trade secret protection will not prevent the use of the protected information by those who independently develop it or by those who acquire it by legitimate means. This means another programmer could independently develop a nearly identical program as long as he or she has not done so based on inappropriate access to the first programmer's source code.

How Software Is Sold

The value of software is in its electronically coded instructions, not in the plastic disc or other tangible media upon which the instructions are stored. Software developers do not sell DVDs or other storage media. What they actually sell is permission to use the programs they have created. This is called *licensing* the software.

Licensing is more like renting a bike than buying one. When you buy a bike, you own it outright and can do what you want with it. When you rent a bike, however, you have the right to use it, but you may have some restrictions on how long and where you can ride it. Similarly, with software, you purchase a license to use the software in a manner specified by the terms and conditions of the license.

Software may be licensed in several ways, including:

Freeware. Obtaining freeware costs nothing. "Free," however, does not necessarily mean unlimited. For example, the license could restrict use to particular users (e.g., personal, noncommercial users), or the license could limit the ability of an individual user to redistribute the software.

Shareware. A software developer may release a program as shareware and expect people who use the program to make a donation or pay a fee. The developer is relying on your honor to help cover the costs of developing the software.

Demo. Demo or trial software will work for a limited time or with limited features. After the purchaser pays a license fee, the programmer provides a code that removes the limitations.

Open Source. An open-source license is a form of freeware. This particular type of license covers the executable program *and* the source code developed by the programmer. This license may come with limitations or restrictions.

Software Piracy

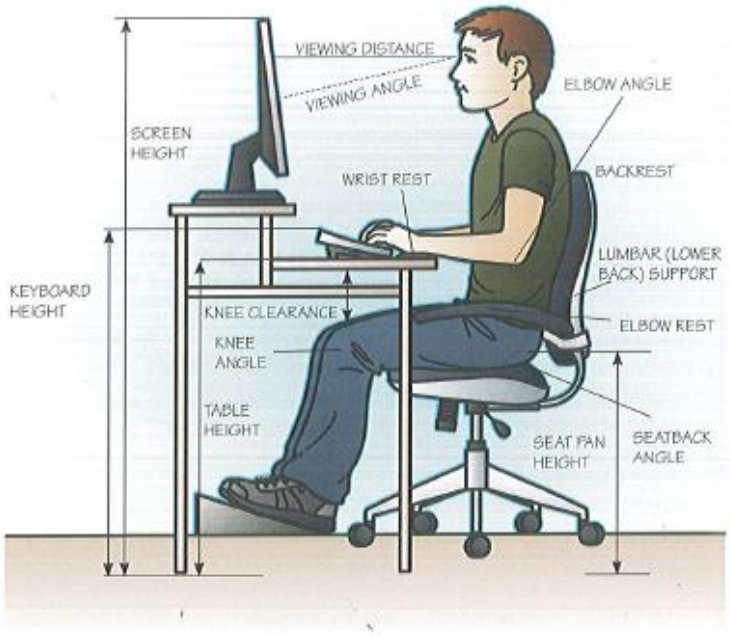
The unauthorized use of another person's intellectual property is called *piracy*. Software piracy is a costly, worldwide problem. According to the Business Software Alliance's 2011 Global Software Piracy Study: "[T]he global piracy rate hovered at 42 percent in 2011 while a steadily expanding marketplace in the developing world drove the commercial value of software theft to $63.4 billion."

The Federal Bureau of Investigation is charged by the U.S. government to investigate piracy and intellectual property theft. This includes ideas, inventions, and creative expressions, such as trade secrets, music, movies, and software. The FBI's Anti-Piracy Warning Seal is used to help "detect and deter criminal violations of U.S. intellectual property laws by educating the public about the existence of these laws and the authority of the FBI to enforce them." **You have probably seen this seal alongside the following text:**

The unauthorized reproduction or distribution of a copyrighted work is illegal. Criminal copyright infringement, including infringement without monetary gain, is investigated by the FBI and is punishable by fines and federal imprisonment.

A Scout Is Trustworthy

Stealing software is like stealing anything else. It's wrong. You would not sneak into a theater to see a movie without purchasing a ticket, or let a friend in through the back door without paying. Similarly, you should only use software you have paid for and not make copies for your friends.



Ergonomics **is the science of equipment design, intended to increase efficiency by reducing operator fatigue and discomfort. For proper ergonomics:**

- **Monitor should be about an arm's length away from eyes (18–24 inches).**
- **Top of screen should be level with eyes so you look slightly down at the monitor.**
- **Knees should be at an open, 90- to 120-degree angle (legs not folded under you).**
- **Elbows at a 90-degree angle.**
- **Wrists resting on support and straight.**
- **Mouse directly in front of elbow (you shouldn't have to reach for it).**
- **Keyboard at elbow height.**
- **Seated with back against backrest, back at 90-degree angle to legs.**
- **Feet flat on floor or on footrest.**
- **Head balanced on neck, not tilted too far back or forward.**

Safety

You might not think that writing programs could cause injuries. But in fact, injuries from programming have much in common with sports injuries and can be prevented using some of the same techniques that athletes use.

The main programming-related injuries are *repetitive stress injuries*, or RSIs. Just as in athletic activities, an RSI occurs when stress is placed on a joint, pulling on the tendons and muscles around the joint. When the stress happens repeatedly, the body does not have time to recover and becomes irritated. The body reacts to the irritation by increasing the amount of fluid in that area to reduce the stress placed on the tendon or muscle.

And just as athletes are trained to do, you can avoid issues like the following by developing good habits.

Carpal tunnel syndrome—swelling inside a narrow "tunnel" formed by bone and ligament in the wrist that can lead to pain, tingling, and numbness; the tunnel surrounds nerves that conduct sensory and motor impulses to and from the hand

Cervical radiculopathy—disk compression in the neck, often caused by repetitive cradling of a phone on the shoulder

Epicondylitis—elbow soreness often called "tennis elbow"

Reflex sympathetic dystrophy—a painful condition marked by dry, swollen hands and loss of muscle control

Tendonitis—tearing and inflammation of tendons connecting bones to muscles

Besides developing RSIs, programmers can become dehydrated, experience back problems, and develop eyestrain and headaches.

Injury Prevention

Proper equipment and preparation are the keys to prevention. Make sure your equipment is set up properly to prevent injuries (see the ergonomics sidebar).

Also be aware that hydration isn't solely for hiking or playing sports—programmers need to stay hydrated too. Four 8-ounce glasses of water per day is a good number. Soda, juices, and other sweet drinks are **not** a substitute for water.

To help prevent eyestrain, program in a well-lit room. Minimize the contrast between your monitor and the rest of the room but make sure there is no glare on the screen.

Proper posture is the key to avoiding back pain and injuries (see the information on ergonomics). And just as when you participate in athletic activities, taking breaks to give your body time to recover and prevent strain is essential.

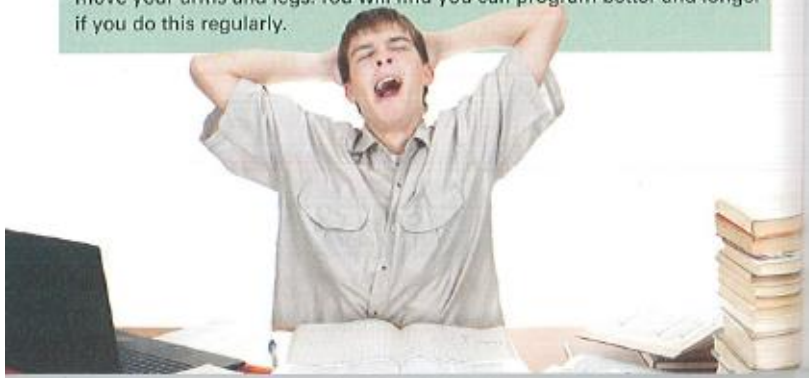
Take a Break

Eye breaks. It's easy to get so focused on programming that you don't move for hours. Take eye breaks—look away from the monitor from time to time, preferably at something more than 20 feet away. This gives your eyes a chance to relax and helps prevent eyestrain.

Typing breaks. Most typing is done in bursts. Rest your hands in a relaxed, flat, straight manner to give them time to recover and prevent RSIs.

Rest breaks. Take a break every 30 minutes or so to give your body a chance to relax. You can use software programs that remind you to take a break so you don't get stuck in a trance staring at your monitor.

Exercise breaks. Get up and stretch, rotate your head and shoulders, move your arms and legs. You will find you can program better and longer if you do this regularly.



First Aid for RSIs

- Apply an ice pack to the injured area to help reduce pain and swelling.
- Use an elastic joint support or wrap the area firmly with an elastic bandage to limit the swelling and to protect the injury. Do not wrap it so tightly that blood circulation is restricted.
- Rest the injured area.
- Take an anti-inflammatory pain reliever as recommended by your physician.
- After 24 hours, heat (hot packs, heating pad, whirlpool) may be applied.
- As symptoms diminish, gently exercise the affected muscles or joints to help relieve remaining tenderness, stiffness, and tingling or numbness.
- If pain is severe or persistent, seek medical attention.

Features of a Human-Friendly Computer Workstation

- Sufficient indirect lighting to prevent eyestrain and glare
- Monitor at eye level to prevent hunching over
- Keyboard at elbow height for arm and shoulder alignment
- Padding in front of keyboard for wrist alignment and relaxation
- Adjustable seatback support for lower back
- Footrest for comfort and stability

Electrical Safety

If you are writing a program on a device that is plugged into the wall for power, you are dealing with potentially deadly electronic circuits. Keep liquids and food away from plugged-in machines, and make sure any cords are neatly stowed to prevent tripping. Be sure the equipment is properly grounded to prevent shock hazards. It's best to unplug the computer when it is not in use, especially during a thunderstorm. All of these suggestions protect you and the computer, too!

Programming Terms

binary code. A numbering system that uses only two digits. In programming, the two digits are a 1 and a 0, which represent turning an electronic circuit on and off.

bit. Short for *binary digit*. The smallest possible unit of information. Each bit represents one electronic switch (or "transistor") in the processor that can be on or off.

byte. A group of eight bits of information.

client. Part of a client-server connection. The client is an application, such as an email program, that runs on a personal computer. That computer is networked to another computer, called a server, which helps the client perform its work, such as sending an email.

coding. Writing instructions using the protocol (rules) of a particular programming language.

commercial software. Software produced commercially and purchased for use.

compiled programming language. A programming language that is translated from a high-level language into another language, usually machine code.

compiler. A program that decodes instructions written in an English-like language and then translates, or compiles, them into machine language.

development environment. A collection of programming tools used for developing, testing, and debugging a program or software application.

embedded processor. A small special-purpose computer usually dedicated to a single task.

ergonomics. Designing and arranging the things people use so that the people and the things interact most efficiently, comfortably, and safely.

flowchart. A diagram that shows step-by-step progression through a procedure or system, usually using connecting lines and basic symbols. To show the flow of work, a process flowchart typically uses a rectangle for a step in the process; a diamond for a *decision point;* a flattened oval for the start or end of the flowchart; and a circle for a connection to another page of the chart.

macro. A sequence of commands in a software application that can be recorded or directly programmed to repeatedly execute the sequence.

malware. Damaging or "malicious" software intended to disrupt a network or a single computer. Types of malware include viruses, worms, and trojans.

operating system. Software that allows the computer to perform basic functions.

portability. A program's ability to run on a variety of processors. Software that is portable (also known as *machine-independent)* does not depend on a particular type of hardware.

program. A set of processor instructions.

programming language. The language used to write instructions that a processor can understand or interpret.

protocol. The rules processors use to communicate with each other.

pseudocode. An outline of a program, written in a form resembling plain English that can be converted into real programming statements.

source code. The original code used to create the program. Depending on the language, this code could be compiled down to machine code or just interpreted as is.

state diagram. A type of diagram used in computer science to illustrate the possible states, or stages of behavior, and all the possible paths a program can use to transition from one state to another.

structured programming. A style of programming aimed at improving the clarity, quality, and development time of a program by using blocks (sections of code that are grouped together) and subroutines.

subroutine. A sequence of program instructions to perform a specific task, written as a unit that can then be used in programs wherever that particular task should be performed. Subprograms may be defined within programs, or separately in libraries that can be used by multiple programs. In different programming languages, a subroutine may be called a function, a routine, or a subprogram.

trojan. A virus or harmful program disguised to look like a useful program, such as a screensaver.

workstation. A terminal or personal computer usually connected to a computer network, or a powerful microcomputer used especially for scientific or engineering work.

worm. A software program that, once installed on a computer, copies itself and sends its copies over a network.

Programming Resources

Scouting Literature

Communication, *Computers*, *Electronics*, and *Robotics* merit badge pamphlets

Visit the Boy Scouts of America's official retail website (with your parent's permission) at http://www.scoutstuff.org for a complete listing of all merit badge pamphlets and other helpful Scouting materials and supplies.

Books

Foxall, James. *Sams Teach Yourself Visual Basic 2012 in 24 Hours*. Sams Publishing, 2012.

Henney, Kevlin. *97 Things Every Programmer Should Know: Collective Wisdom From the Experts*. O'Reilly Media, 2010.

Horstmann, Cay S. *C++ for Everyone*, 2nd ed. Wiley, 2010.

Newsome, Bryan. *Beginning Visual Basic 2012*. Wrox, 2012.

Sharp, John. *Microsoft Visual C# 2012*. Microsoft Press, 2013.

Watson, Karli, Jacob Vibe Hammer, Jon Reid, Morgan Skinner, et al. *Beginning Visual C# 2012 Programming*. Wrox, 2012.

Organizations and Websites

Android
Tutorials for Android app building
Website: http://developer.android.com/training/index.html

Code.org
Free tutorials and introductions to programming
Website: http://www.code.org

HowToStartProgramming.com
Beginner information about programming for Visual Basic and PHP
Website: http://howtostartprogramming.com

The best place to start your programming journey is with the companion website for this merit badge, www.boyslife.org/programming. There you will find many examples and free resources appropriate for Scouts. You will be up and running quickly and be able to find what you need to fulfill the Programming merit badge requirements.

InterConnecting Automation Inc.
Free access to Scouts (send them a note); learn about PLCs (programmable logic controllers)
Website: http://www.interconnectingautomation.com

Learn C++
Free tutorials and other resources on how to program in C++
Website: http://learncpp.com

Learnpython.org
Interactive Python tutorial
Website: http://www.learnpython.org

Oracle Corporation
Java tutorials
Website: http://docs.oracle.com/javase/tutorial

Robotics Academy of Summer Learning
From the Carnegie Mellon Robotics Academy, animation, robotics, web design, game design, and more
Website: http://www.cs2n.org

Scratch
Good, free examples of programs
Website: http://scratch.mit.edu

U.S. Copyright Office
Website: www.copyright.gov–Copyright Office

U.S. Patent and Trademark Office
Website: http://www.uspto.gov

W3schools.com
Tutorials for all web design programming tools
Website: http://www.W3schools.com

Acknowledgments

The Boy Scouts of America thanks the following members of the Programming Merit Badge Development Team, who diligently worked to develop the merit badge requirements and content for this pamphlet.

Rick Folea—Eagle Scout; Programming merit badge lead; AutomationDirect.com; *Robotics* merit badge pamphlet co-author

Special thanks to Automation Direct.com, an industry leader in factory automation, for providing the services of Rick Folea. He coordinated the development of the Programming merit badge, and AutomationDirect.com supported the launch of the merit badge at the 2013 National Scout Jamboree with equipment, supplies, and personnel.

Celeste Rance—Software development manager, Pearson

David Kerven, Ph.D., J.D.—Associate professor of Information Technology, Georgia Gwinnett College

Robert J. Caruso, CISSP—Information security architect, Battelle Memorial Institute; U.S. CyberPatriot mentor and lifelong Scouter; BSA Cyber Chip co-developer

Laszlo Hideg, Ph.D., P.E.—Electrical engineering components, electric motor controls, Chrysler LLC; *Robotics* merit badge pamphlet co-author; *FIRST*® Robotics competition judge, volunteer, and mentor of Team 33, "The Killer Bees"

James Francisco, Ph.D.—Associate faculty, School of Advanced Studies, University of Phoenix, software quality engineering, software test automation

Steve Wincor—Chief systems architect, Lockheed Martin; BSA Nova Awards mentor; Cyber Chip codeveloper

Curtis Heisey—Eagle Scout; Master of Arts in physics; software engineer, MIT Lincoln Laboratory; coach, Team America Rocketry Challenge

We would also like to acknowledge James Trobaugh, Chip McDaniel, Eagle Scout Christopher Hideg, Talia Heisey, and Theresa Folea for their assistance in developing the code examples used throughout this book and on the companion website. Thanks to Eagle Scout Christopher Folea for graphics support and the cover design concept. We also appreciate VEX Robotics and RadioShack Corporation for their assistance with providing hardware in support of this project.

Photo and Illustration Credits

AutomationDirect.com, courtesy—pages 39, 41 *(all)*, 42 *(all)*, and 43 *(both)*

AutomationDirect.com/Christopher Folea, courtesy—page 62 *(bottom)*

Frank da Cruz, courtesy—page 12 *(loom)*

Federal Bureau of Investigation, courtesy—page 85

Rick Folea, courtesy—page 47

Christopher Hideg and Laszlo Hideg, courtesy—page 10

LEGO, the LEGO logo, MINDSTORMS, and the MINDSTORMS logo are trademarks of the LEGO Group. ©2011 The LEGO Group. Colors and decorative designs may vary.—page 45 *(top)*

Shutterstock.com—cover (*smartphone*, 86181757 ©Oleksiy Mark; *laptop*, 57987106 ©ra2studio); pages 3 (25689409 ©Mauro Saivezzo), 6 (66456049 ©rzymuR), 8 (91594358 ©VLADGRIN), 9 (83701204 ©VLADGRIN), 12 (*background*, 106145765 ©Essl), 15 (84418210 ©URRRA), 21 (1690528 ©Brian McEntire), 21 (*Hello World*, 25689409 ©Mauro Saivezzo), 22 (13328182 ©Vilmos Varga), 24 (49830424 ©maxstockphoto), 32 (*background*, 132534890 ©morrison 77; *coffeemaker*, 77807080 ©jcjgphotography; *scoreboard*, 57422011 ©Nelson Marques; *clock*, 59859022 ©toots; *navigation screen*, 53194405 ©Rob Bouwman; *remote*, 50411152 ©Santiago Cornejo), 34 (*teenager*, 85803271 ©Ljupco Smokovski; *electronic tablet*, 124122853 ©Pixsooz; *couple studying*, 124813948 ©Goodluz; *boy*, 112139366 ©Roberto Caucino), 37 (38258086 ©Vartanov Anatoly), 38 (*top*, 56710474 ©Rainer Plendl; *bottom*, 68207422 ©emel82), 44 (79049326 ©Paul Fleet), 45 (*bottom*, 8051164 ©Stavchansky Yakov), 49 (106142042 ©zeber), 50 (*bottom*, 96151733 ©razihusin), 56 (*top*, 57406054 ©Gala; *drops*, 103324052 ©Maen CG; *RBG example*, 46943026 ©DVARG), 57 (*top*, 129275894 ©Oleg Zhevelev; *bottom*, 132490523 ©Algol), 59 (*top*, 12933343 ©Keith Muratori; *bottom*, 27668230 ©Monkey Business Images), 60 (*e-book*, 96467273 ©Adam Radosavljevic; *GPS unit*, 2909132 ©Matt Cooper; *teenager*, 54864916 ©Monkey Business Images; *watch*, 99100199 ©Venus Angel; *stopwatch*, 17846359 ©Joe Belanger; *music player*, 104096348 ©Georgejmclittle), 61 (*bottom*, 123176794 ©Christian Delbert), 64 (*top*, 10333774 ©Regien

Paassen; *bottom*, 1358787 ©Thomas D Dittmer), 65 (*top*, 110260040 ©Verticalarray; *bottom*, 129399833 ©Yuri Arcurs), 67 (36345919 ©ID1974), 68 (*top*, 135018554 ©Saulius L.; *bottom*, 48660928 ©Norebbo), 69 (44433292 ©manfredxy), 71 (119998051 ©tomasso79), 72 (60891964 ©Pedro Miguel Sousa), 73 (*top*, 125882543 ©Maksim Kabakou; *bottom*, 71425324 ©Monkey Business Images), 74 (*top*, 103378928 ©Maksim Kabakou; *bottom*, 130473632 ©Gajus), 76 (*top*, 113190256 ©Shutter_M; *bottom*, 81309157 ©auremar), 77 (94945093 ©Hugo Felix), 78 (121321996 ©Login), 80 (*planet with red arrow*, 132959393 ©Milos Dizajn; *outer space fantasy*, 79793797 ©malamalama), 82 (*bottom*, 86691250 ©arka38), 83 (*background*, 27596377 ©Benjamin Haas; *man*, 107516819 ©Uros Zunic), 84 (135578990 ©faysal), 887 (137806895 ©sixninepixels), 88 (*top*, 100985494 ©violetblue; *bottom*, 121132099 ©Andrey Shadrin), 90 (129334397 ©Rafal Olechowski), and 92 (114134812 ©jimmy), 2013 Used under license from Shutterstock.com

Wikipedia.org/Evan-amos, courtesy—page 60 *(top)*

Wikipedia.org/Mcbort, courtesy—page 61 *(top)*

Wikipedia.org/National Oceanic and Atmospheric Administration/ Hydrometeorological Prediction Center, courtesy—page 62 *(top)*

Wikpedia.org/U.S. Dept. of Energy/ LANL, courtesy—page 19

All other photos and illustrations not mentioned above are the property of or are protected by the Boy Scouts of America.

John McDearmon—page 86