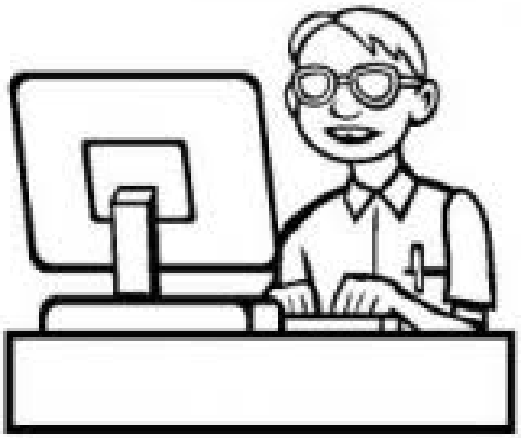
SECTION 6. JOBS IN ICT /PROGRAMMING



LEARNING OBJETIVES

Study basic concepts in programming

Learn and use vocabulary connected with programming

Practice the use and pronunciation of the -ed form verbs

Discuss the personal qualities and professional skills needed for a job in ICT

Contenido

1. INTRODUCTION

Most people on ASIR are either studying for their first job in IT, or else trying to improve their current IT career. If this is the case with you, well then, this unit should really help. Sometimes the hardest part of meeting a goal is to properly define what you are trying to accomplish in the first place. In this section we will discuss the top IT job positions available around the world right now.

There are several things to keep in mind when determining what field of IT to go into. Keep an eye on job web sites such as infojobs to see which jobs are most in-demand. Keep in mind that for many jobs described below, there are several levels of positions available. For instance, there are "junior", "senior", and "lead" software developer positions available. You probably can't start out you career as a lead developer. You have to know your own limits.

Be honest with yourself. If you don't have previous experience, good contacts, or a good degree from a well-known university, you will be more successful in getting a lower-level job. Also, find out what the job you are applying for typically pays in your area. If you are young, living in a financially depressed area, or really need a job, keep your salary expectations a bit lower than the average. This will make your chances much higher than normal to get hired. Once you have "job experience" then you will be in a good position to ask for more money.

Everyone who works hard deserves a raise every year. How do you show your IT manager that you are a good performer? Easy. Show up on time, be dependable, be active in the meetings, and always do a little bit more than is asked of you. Also, equally important is to be well-liked by members of your team.

Learn something new every day

IT is an area where people are judged largely by how much they know. If money and a high job position are important to you, you can quickly raise your level by telling your manager that you want harder tasks and more responsibility. IT Managers normally love it when employees ask for more responsibility. When you meet with your manager, set goals for yourself and meet or exceed those goals. Here are some things you can do to increase your worth to your company:

- learn a new programming language
- take a certification such a Microsoft, Linux Professional Institute, or Cisco
- study to be a ScrumMaster or another type of project manager.

Meeting set goals can have beneficial results when it's time to renegotiate salaries, survive a round of layoffs, or get a promotion.

Be courteous, helpful, and respectful to others

In a good IT department, the engineers are known for sharing knowledge and helping each other. In bad IT departments, the engineers are secretive and hide knowledge. How can everyone get better if some people are selfish with what they know? Information wants to be free. You must set it free. Despite the fact that I have been to several universities, graduate school, and have collected many IT certifications, I have still learned much more about IT from my fellow engineers than from all my higher education combined. So, my advice is to be kind and respect your fellow IT staff. They are your family for eight hours every day, forty hours every week!

When you start a new job, realize how some people are nice to you and some people ignore you. Which kind of person do you want to be? When you get a new junior team member, try to help them and include them in decisions. Make sure they have someone to eat lunch with. If you party after work with your co-workers, invite new employees with you. Being nice to new people can have many rewards, both emotionally and financially.

Back in the 90's we used to have a saying, "Think globally, act locally." What this means is that just by being nice and pleasant yourself, you can make the whole world a more nice and pleasant place as well.

Have your own mind and your own opinions

State your opinions in meetings and give good reasons and facts to back up your opinions. But don't be stubborn or insistent if things don't go your way. And whatever you do, please don't be passive-aggressive! Passive-aggressive behavior is when you think something bad about a person or an idea, and then you talk badly behind someone's back (when that person is not around). This is very destructive behavior to both yourself and your IT department.

2. JOBS IN ICT

Okay, now we are ready now to investigate some popular IT job positions. I will rate the following jobs based on the following criteria: position, qualities, salary. I will also include some notes. These are subjective opinions. Some are even intentionally funny.

CTO (Chief Technical Officer), CIO (Chief Information Officer)

Position: Very High

Qualities: Business savvy, technical mindset, good people skills

Average Salary: $150,000

Notes: These jobs are highly competitive and usually political, so your chances are low. Sorry.

Enterprise Architect

Position: High

Qualities: Good technical, business, and design skills

Salary: $100,000

Notes: Responsible for all solutions that work; not responsible for ones that don't work

IT Manager

Position: Medium-High

Qualities: Detail oriented, punctual, critical, supportive

Salary: $70,000

Notes: They always seem to be working

Technical Writer

Position: Medium

Qualities: Excellent writing skills, good technical mind

Salary: $50,000

Graphic Designer

Position: Low-Medium

Qualities: Excellent drawing and illustration skills, good color matching and artistic qualities

Salary: $50,000

Notes: Not as 'square' as the rest of the IT department. All good designers seem to have tattoos, piercings, and a fashion sense.

Software Developer

Position: Medium

Qualities: Creative, persistent, insatiable thirst for knowledge

Salary: $70,000

Notes: Companies have a lot of developers compared to other positions listed. Therefore, your chances of becoming a developer are good if you have the skills and more importantly the desire.

Project Manager

Position: Medium

Qualities: Cooperation, leadership, and organization skills

Salary: $60,000

Database Developer / Database Administrator

Position: Medium-High

Qualities: Detail-oriented, high business knowledge

Salary: $80,000

IT Security Manager

Position: High

Qualities: Military outlook on life, defensive, pro-active

Salary: $70,000

System Administrator

Position: Medium-High

Qualities: God complex, often eat fast food and drink a lot of soda

Salary: $75,000

Notes: Never anger a sysadmin! Why? They have access to everything in the company.

Software Tester

Position: Low-Medium

Qualities: Detail-oriented, persistent, curious

Salary: $40,000

Notes: Testers play a vital role in software development that cannot be understated. If you don't have a single tester on your team, you are probably in trouble.

IT Support Engineer

Position: Low (except when someone needs help fixing their computer, then it's really high)

Qualities: Must be good at dealing with technically incompetent people

Salary: $35,000

3. Famous People in the History of IT

Almost everyone uses computers these days for everything from shopping to working to playing games. But have you ever stopped to think about where all this amazing technology came from? Who invented it all? Well, behind every company, programming language or piece of software, there is a person - or sometimes a team of people - who turned ideas into reality. We've all heard of Bill Gates, the founder of Microsoft and one of the richest men in history. Equally famous is Steve Jobs, the person who, along with Steve Wozniak, started Apple computers. However, there are hundreds of other people, from early pioneers to later geniuses, who aren't as well-known but who deserve recognition for the work they did in advancing the world of computing.

One of the first people to conceive of computers was Charles Babbage, an English mathematician and analytical philosopher who drew up plans for the first programmable computer called the Difference Engine. George Boole came up with a way of describing logical relations using mathematical symbols - now called Boolean logic - that is the basis of all modern computer processes. Vannevar Bush first proposed an idea in 1945 he called 'memex', which we now know as 'hypertext'. Another notable figure in early computing was Alan Mathison Turing, an Englishman known as the "father of computer science". He invented the Turing Test, which is a way to find out if a computer is acting like a machine or a human. Another English computer scientist, Edgar Frank Codd, is known for inventing the "relational" model for databases, a model which is still in use today.

As computing became more complicated, people needed a way to make it easier to tell computers what to do - in other words, they needed ways to program the computers. These computer instruction systems became known as computer, or programming, languages. FORTRAN, the first widely used high-level programming language, was invented by an American computer scientist, John Warner Backus. Other notable North American inventors of programming languages include Dennis Ritchie, author of the C programming language, Larry Wall, creator of Perl, and Canadian James Gosling, known as the father of Java. Two men from Denmark are responsible for writing two other famous programming languages. Bjarne Stroustrup came up with C++ and Rasmus Lerdorf devised PHP. Dutchman Guido van Rossum wrote the Python programming

language, while the Japanese computer scientist, Yukihiro Matsumoto, made a language called Ruby.

One of the uses of programming languages is to create operating systems, which are essentially sets of instructions that allow computers to function. The most widely-used operating system in the world is Microsoft Windows, but there are other powerful ones that exist, such as Unix, created by Ken Thompson and his team at AT&T in 1969, and Linux, written by Linus Torvalds in 1991.

Microsoft, of course, is the largest software company in the world, but there is another company, Intel, that is equally important when it comes to hardware. Intel was started by several people who are now legends in the computer world, including Robert Noyce and Gordon Moore. Moore is also famous for coming up with Moore's Law, which predicts the rapid increase of computer technology over time. Intel expanded rapidly during the 1980s and 1990s when a man named Andy Grove was in charge of the company.

Other notable figures in the evolution of the computer industry are Ralph Baer, inventor of the first home video game console, Seymour Cray, for many years the manufacturer of the world's fastest supercomputers, Richard Stallman, founder of the free software movement called GNU, and Tim Berners-Lee, the man who created the basis for the World Wide Web.

Through their creativity and hard work, all of these people contributed to shaping what we now experience as Information and Computer Technology. Every time you boot up a computer, play a video game or surf the Internet, try to remember the individuals who made these wonders possible.

4. PROGRAMMING SOFTWARE

Programming software is the type of software that is not used by end-users. It is not for you unless of course, you are a programmer. Programming software are programs that are used to write, develop, test, and debug other software, including apps and system software. For someone who works at a bespoke software development company, for example, this type of software would make their life easier and efficient.

Programming software is used by software programmers as translator programs. They are facilitator software used to translate programming languages (i.e., Java, C++, Python, PHP, BASIC, etc) into machine language code. Translators can be compilers, interpreters and assemblers. You can understand compilers as programs that translate the whole source code into machine code and execute it. Interpreters run the source code as the program is run line by line. And assemblers translate the basic computer instructions – assembly code – into machine code.

Different programming language editors, debuggers, compilers and Integrated Development Environment (IDE) are an example of programming software. For example:

- Eclipse – a Java language editor
- Coda – programming language editor for Mac
- Notepad++ – an open-source editor for windows
- Sublime Text – A cross-platform code editor for Mac, Windows, and Linux

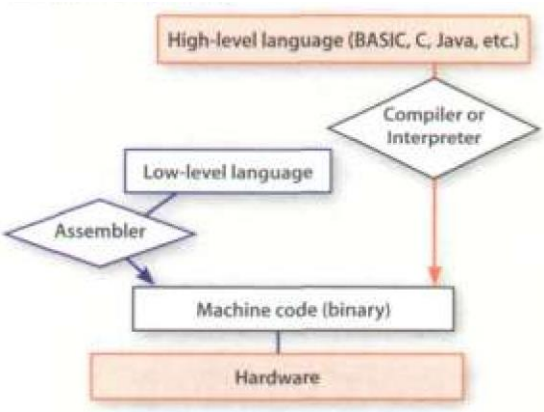
Unfortunately for us, computers can't understand spoken English or any other natural language. The only language they can understand directly is machine code, which consists of 1s and 0s (binary code).

Machine code is too difficult to write. For this reason, we use symbolic languages to communicate instructions to the computer. For example, assembly languages use abbreviations such as ADD, SUB, MPY to represent instructions. The program is then translated into machine code by a piece of software called an assembler. Machine code and assembly languages are called low-level languages because they are closer to the hardware. They are quite complex and restricted to particular machines. To make the programs easier to write, and to overcome the problem of intercommunication between different types of computer, software developers designed high-level languages, which are closer to the English language.

Here are some examples:

- FORTRAN was developed by IBM in 1954 and is still used for scientific and engineering applications.
- COBOL (Common Business Oriented Language) was developed in 1959 and is mainly used for business applications.
- BASIC was developed in the 19605 and was widely used in microcomputer programming because it was easy to learn. Visual BASK is a modern version of the old BASIC language, used to build graphical elements such as buttons and windows in Windows programs.
- PASCAL was created in 1971. It is used in universities to teach the fundamentals of programming.
- C was developed in the 1980s at ATAT. It is used to write system software, graphics and commercial applications. C++ is a version of C which incorporates object-oriented programming: the programmer concentrates on particular things (a piece of text, a graphic or a table, etc.) and gives each object functions which can be altered without changing the entire program. For example, to add a new graphics format, the programmer needs to rework just the graphics object. This makes programs easier to modify.
- Java was designed by Sun In 1995 to run on the Web. Java applets provide animation and interactive features on web pages.

Programs written in high-level languages must be translated into machine code by a compiler or an interpreter. A compiler translates the source code into object code - that is, it converts the entire program into machine code in one go. On the other hand, an interpreter translates the source code line by line as the program is running.



It is important not to confuse programming languages with mark-up languages, used to create web documents. Mark-up languages use instructions, known as mark-up tags, to format and link text files. Some examples Include:

- HTML, which allows us to describe how information will be displayed on web pages.
- XML, which stands for Extensible Markup Language. While HTML uses pre-defined tags, XML enables us to define our own tags; it is not limited by a fixed set of tags.

```
<xml>
< name> Andrea Finch </name>
< homework> Write a paragraph describing the C language </homework>
</xml>
```

- VoiceXML, which makes Web content accessible via voice and phone. VoiceXML |s used to create voice applications that run on the phone, whereas HTML is used to create visual applications (for example, web pages).

5. Top 10 Most Popular Programming Languages

There are hundreds of programming languages in use today. How can you know which one to learn first? How do you know which ones are the best for your IT field of choice? Well, I can't answer that question for you. But why not start by learning one of the top 10 most popular ones? That way you will always be able to get a job in the IT industry.

Learning a programming language is not easy, but it can be very rewarding. You will have a lot of questions at first. Just remember to get help when you need it! You can find out the answer to almost everything on Google nowadays.... so there is no excuse for failure. Also remember that it takes years to become an expert programmer. Don't expect to get good overnight. Just keep learning something new every day and eventually you will be competent enough to get the job done ;)

This article covers the top 10 most popular programming languages as ranked by Tiobe.com in June 2009. I have added some general reviews and comments about each language they listed. Remember these are my own personal opinions. Other IT professionals might have different opinions.

1. Java

Java uses a compiler, and is an object-oriented language released in 1995 by Sun Microsystems. Java is the number one programming language today for many reasons. First, it is a well-organized language with a strong library of reusable software components. Second, programs written in Java can run on many different computer architectures and operating systems because of the use of the JVM (Java virtual machine). Sometimes this is referred to as code portability or even WORA (write once, run anywhere). Third, Java is the language most likely to be taught in university computer science classes. A lot of computer science theory books written in the past decade use Java in the code examples. So, learning Java syntax is a good idea even if you never actually code in it.

Java Strengths: WORA, popularity

Java Weaknesses: Slower than natively compiled languages

2. C

C is a compiled, procedural language developed in 1972 by Dennis Ritchie for use in the UNIX operating system. Although designed to be portable in nature, C programs must be specifically compiled for computers with different architectures and operating systems. This helps make them lightning fast. Although C is a relatively old language, it is still

widely used for system programming, writing other programming languages, and in embedded systems.

Strengths: Speed

Weaknesses: Memory management can be difficult to master

3. C++

C++ is a compiled, multi-paradigm language written as an update to C in 1979 by Bjarne Stroustrup. It attempts to be backwards-compatible with C and brings object-orientation, which helps in larger projects. Despite it's age, C++ is used to create a wide array of applications from games to office suites.

Strengths: Speed

Weaknesses: C++ is older and considered more clumsy than newer object-oriented languages such as Java or C#.

4. PHP

PHP uses a run-time interpreter, and is a multi-paradigm language originally developed in 1996 by Rasmus Lerdorf to create dynamic web pages. At first it was not even a real programming language, but over time it eventually grew into a fully featured object-oriented programming language. Although PHP has been much criticized in the past for being a bit sloppy and insecure, it's been pretty good since version 5 came out in 2004. It's hard to argue with success. Today, PHP is the most popular language used to write web applications. Even English 4 IT, the program you are currently using, is written in PHP ;)

Strengths: Web programming, good documentation

Weaknesses: Inconsistent syntax, too many ways to do the same thing, a history of bizarre security decisions

5. VB (or Visual Basic)

Visual Basic is an interpreted, multi-paradigm language developed by Microsoft Corporation for the Windows platform. It has been evolving over the years and is seen as a direct descendant of Microsoft's old BASIC from the 1970's. Visual Basic is a good language for scripting Windows applications that do not need the power and speed of C#.

Strengths: None.

Weaknesses: Only runs in Windows

6. Python

Python is an interpreted, multi-paradigm programming language written by Guido van Rossum in the late 1980's and intended for general programming purposes. Python was not named after the snake but actually after the Monty Python comedy group. Python is characterized by its use of indentation for readability, and its encouragement for elegant code by making developers do similar things in similar ways. Python is used as the main programming choice of both Google and Ubuntu.

Strengths: Excellent readability and overall philosophy

Weaknesses: None

7 C#

C# is a compiled, object-oriented language written by Microsoft. It is an open specification, but rarely seen on any non-Windows platform. C# was conceived as Microsoft's premium language in its .NET Framework. It is very similar to Java in both syntax and nature.

Strengths: Powerful and pretty fast

Weaknesses: Only really suitable for Windows

8. JavaScript

JavaScript is an interpreted, multi-paradigm language. A very strange one too. Despite it's name, it has nothing whatsoever to do with Java. You will rarely, if ever, see this language outside of a web browser. It is basically a language meant to script behaviors in web browsers and used for things such as web form validation and AJAX style web applications. The trend in the future seems to be building more and more complex applications in JavaScript, even simple online games and office suites. The success of this trend will depend upon advancements in the speed of a browser's JavaScript interpreter. If you want to be correct, the real name of this programming language is ECMAscript, although almost nobody actually calls it this.

Strengths: it's the only reliable way to do client-side web programming

Weaknesses: it's only really useful in a web browser

9. Perl

Perl is an interpreted, multi-paradigm language written by Larry Wall in 1986. It is characterized by a somewhat disorganized and scary-looking syntax which only makes sense to other PERL programmers ;) However, a lot of veteran programmers love it and use if every day as their primary language. 10 years ago, Perl was more popular than it is

today. What happened? A lot of newer programmers and even old Perl programmers (such as myself) have switched to other languages such as PHP, Python, and Ruby. Perl is perhaps still the best language for text processing and system administration scripting. I personally do not recommend it however as a primary programming language.

Strengths: text processing and system administration

Weaknesses: strange syntax, and perhaps too many ways to do the same thing

10. Ruby

Ruby is an interpreted, object-oriented language written by Yukihiro Matsumoto around 1995. It is one of the most object-oriented languages in the world. Everything is an object in Ruby, even letters and numbers can have method calls. It's a great language to learn if you love objects. The only negative is that it's love of object-orientation makes it a bit slow, even for an interpreted language.

Strengths: Perhaps the world's most object-oriented language

Weaknesses: its superior object model comes at a price... namely speed

Okay! Those are the top 10 programming languages in use today and some personal comments about them. Remember that opinions are like noses, everyone has one and they all smell ;) If you disagree, please feel free to email me or write your own opinions on the forum.

6. LANGUAGE WORK

The infinitive

The infinitive with to is used in the following ways.

- To express purpose
  - We *use* symbolic languages to communicate Instructions to the computer. (In order to communicate)

    Not: ... for to communicate
- After adjectives
  - *BASIC was widely used in the past because 1 was easy to learn*
  - *Machine code is too difficult to write. (not easy enough to write)*
- After certain verbs (e.9. afford, demand, plan, agree, expect, promise. appear, hope. refuse, arrange. learn, try, decide, manage)
  - *A lot of companies are now trying to develop voice applications for web access.*
- After the object of certain verbs (e.g. advise, encourage. allow, expect. tell, ask, invite, want, enable, order, warn)
  - *HTML allows us to describe how information will be displayed on web pages.*

The bare infinitive (without to) is used in the following ways.

- After modal verbs (e.g. can, could, may. might. will, would, must, should)
  - *Unfortunately, computers can't understand spoken English.*
  - *High-level languages must be translated into machine code.*
- After the object with the verbs make and let
  - *Programs* make computers perform specific tasks.

The -ed form

We use the -ed form in the following ways:

- To make the past simple (affirmative) of regular verbs
  - *Sun Microsystems developed Java in 1995.*
  - Remember that not all verbs in the past simple end in -ed. See section 5 for a list of irregular verbs.
- To make the past participle of regular verbs
  - *Flash is used to create animation.*
- To make the adjectival form of some verbs
  - Java applets let you watch animated characters.
- The -ed is pronounced as:
  - /t/ after voiceless sounds: "-f ", "-k ", "-p ", "-s ", "-sh ", "-ch ", o "-th "
  - /d/after voiced sounds: "-b", "-g", "-j", "-l", "-m", "-n", "-z", "-v", "-ng", y "-th"
  - /id/ after /t/ or /d/ (e.g. interpreted, multi-threaded)

Present perfect simple

- We form the present perfect simple with have/has + past participle.
    - *We used Microsoft Access for many years.*
    - *I haven't used Microsoft Access for years.*
- We use this tense to talk about:
  - States that started in the past and continue to the present.
    - *Since 2006, I've been a computer operator for PromoPtint.*
  - Past actions that continue to the present, where we put an emphasis on quantity (how many).
    - *I have designed four programs in COBOL.*
  - Personal experiences, especially with ever and never.
    - *Have you ever worked with databases?*
    - *I've never worked with databases.*

Present perfect continuous

- We form the present perfect continuous with have/has been + present participle.
    - *Since January I've been writing programs in C.*
- We use this tense to talk about:
    - Actions which started in the past and are still happening.
        - *For the last three years I've been working as a software engineer for Intelligent Software.*
    - Past actions that continue to the present, where we put an emphasis on duration (how long).
        - *She's been working all morning.*

Contrast with the past simple

- We use the past simple to talk about events that happened at a specific time in the past that are now finished:
    - *I graduated in May 2003.*
    - *Not: I have graduated in ...*
    - *I stayed in this job until March 2004.*
    - *Two years ago, | spent three months in Spain.*

For, since, ago, until

- We use for to refer to a period of time.
    - *I've lived in Liverpool for five years.*
- We use since to refer to a point in time.
    - *I've been unemployed since May 2005.*
- We use ago with the past simple to say when something happened. We put ago after the time period.
    - *I got married five years ago.*
- We use until to mean up to a certain time
    - *I stayed at high school until I was 18.*

7. SELF-ASSESMENT

1) ***'portability'***
   a) *any programming language that is based on a step-by-step approach to solving a problem*
   b) *a program that reads a high-level programming language, converts it into machine code, and then immediately runs that code*
   c) *a measure of how easily programs can be moved to a new system without having to make any changes.*
2) ***'multi-paradigm language'***
   a) *a program that takes human readable code and turns it into machine readable code for running at a later time*
   b) *a programming language that supports both procedural and object-oriented programming philosophies*
   c) *rules governing the structure of a programming language*
3) ***'compiler'***
   a) *a program that takes human readable code and turns it into machine readable code for running at a later time*
   b) *a program that reads a high-level programming language, converts it into machine code, and then immediately runs that code*
   c) *any programming language optimized for modelling real-world objects and concepts*
4) ***'syntax'***
   a) *a programming language that supports both procedural and object-oriented programming philosophies*
   b) *rules governing the structure of a programming language*
   c) *any programming language optimized for modelling real-world objects and concepts*
5) ***'elegant code'***
   a) *concise, clean, and clear code which allows other developers to understand and extend it*
   b) *a program that takes human readable code and turns it into machine readable code for running at a later time*
   c) *any programming language optimized for modelling real-world objects and concepts*

6) 'procedural language'

a) *a programming language that supports both procedural and object-oriented programming philosophies*

b) *any programming language that is based on a step-by-step approach to solving a problem*

c) *a program that takes human readable code and turns it into machine readable code for running at a later time*

7) 'Python'

a) *a high-level, interpreted programming language written by Larry Wall in 1986 and typically used for a wide variety of programming tasks including system administration*

b) *a popular web page scripting language created by Brendan Eich at Netscape to provide client-side interactivity in Web pages.*

c) *a high-level, interpreted programming language developed by Guido van Rossum at CWI in the Netherlands*

8) 'Java'

a) *a software framework by Microsoft which executes code via a virtual machine*

b) *a high-level, compiled, object-oriented programming language developed by Sun Microsystems and now owned by Oracle.*

c) *a high-level, interpreted programming language written by Rasmus Lerdorf in 1995 and aimed mainly at web developers creating dynamic applications*

9) 'object-oriented language'

a) *any programming language optimized for modeling real-world objects and concepts*

b) *a program that reads a high-level programming language, converts it into machine code, and then immediately runs that code*

c) *rules governing the structure of a programming language*

10) 'interpreter'

a) *a programming language that supports both procedural and object-oriented programming philosophies*

b) *rules governing the structure of a programming language*

c) *a program that reads a high-level programming language, converts it into machine code, and then immediately runs that code*

ANSWERS

1	2	3	4	5	6	7	8	9	10
C	B	A	B	A	B	C	B	A	C