PROGRAMACIÓN ESTRUCTURADA

AUTOR: WALTER MADRIGAL CHAVES

NOVIEMBRE: 2020

San Marcos

Contenido

INTRODUCCIÓN

La programación es una herramienta fundamental en el desarrollo de soluciones informáticas para las organizaciones y la sociedad en general. Igual que todas las herramientas que se emplean en la actualidad, esta presenta una evolución significativa donde podemos ver diferentes modelos que permiten que se desarrollen software y plataformas de alto nivel.

Inicialmente se desarrollaron programas sencillos que eran fáciles de entender y comprender porque se utilizaba la programación estructurada donde los procesos se creaban mediante sentencias secuenciales.

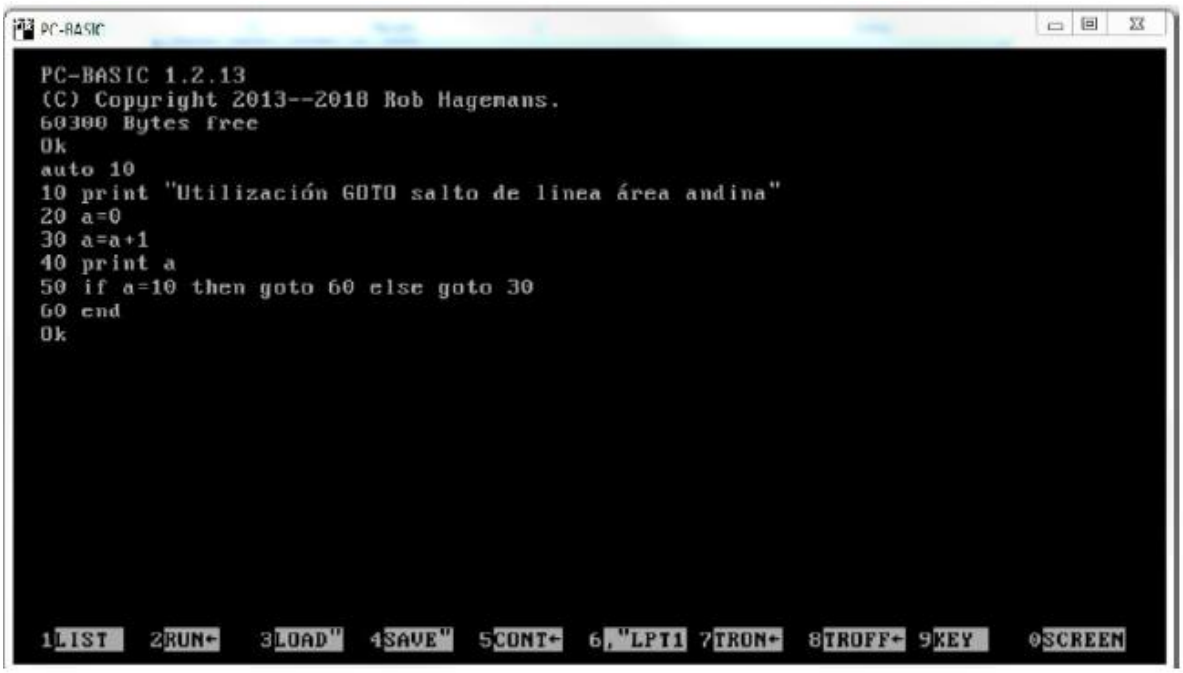
La dinámica del comercio y la globalización de mercados obligó a que todas las organizaciones fueran competentes, lo que llevó a que estas se tecnificaran y abrió espacios para el desarrollo de productos de software a una velocidad impresionante, dinamizó las técnicas y estrategias para el desarrollo de aplicaciones de forma más ágil lo que generó la aparición de nuevos métodos para el desarrollo de soluciones como la programación orientada a objetos (POO).

PRECEDENTES DE LA PRORAMACIÓN ESTRUCTURADA

Antes de que la programación estructurada hiciera su aparición en el contexto de codificación de un programa, se desarrollaron lenguajes de programación como el Basic, GW_Basic o básico. Dichos lenguajes presentaron inconvenientes, como es el caso del código de la programación espagueti, que inicia en un punto y no se sabe dónde termina, debido al empleo de tantos saltos con la instrucción GOTO.

A continuación, se presenta un ejercicio en una plataforma de programación de los años 70: el GW_BASIC, que se entiende como un programa de secuencias con salto de línea para hacer un ciclo o proceso repetitivo que inicia en uno, se incrementa de uno en uno y termina cuando llega a 10.

Imagen 1 Código en GW_BASIC



Fuente:

En el ejercicio se observa una secuencia de instrucciones que se enumeran de 10 en 10, se explica el salto de línea con la instrucción GOTO para generar un

proceso repetitivo, lenguajes estructurados que evolucionaron hasta convertirse en interacciones, ciclos o bucles como el for, while y do while de la actualidad.

En este lenguaje se comenzaron a desarrollar los primeros videojuegos como frog y donkey por mencionar algunos; hay que recordar que en ese tiempo era un lenguaje muy básico y limitado. Se ejecuta el programa pulsando la tecla de función F2.

Imagen 2 Salida de resultado en GW_Basic

Se puede visualizar en la imagen, la salida de resultados de 1 en 1 hasta 10. El GOTO salta validando una condición y se regresa a la línea n de nuestro programa, lo anterior en la actualidad se hace con una instrucción For.

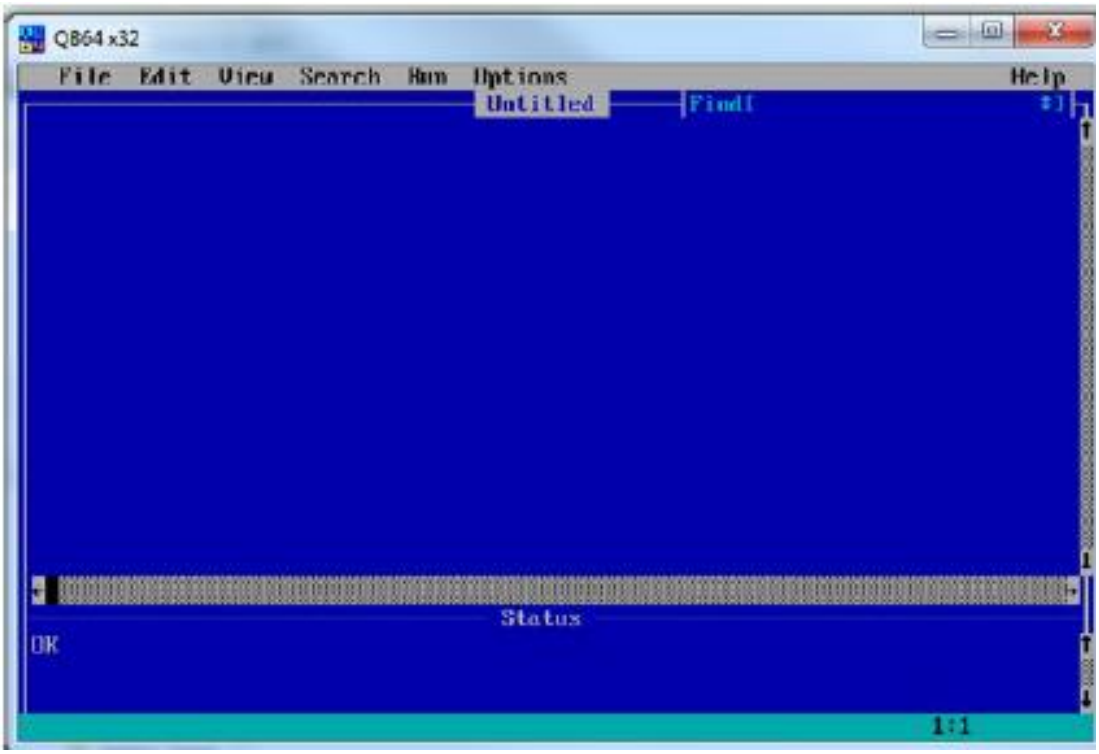
LECTURAS DE DATOS POR TECLADO

En todos los lenguajes de programación se deben leer datos por teclado, para ellos se utilizan las variables, estas son cargadas con información por muchos medios como una entrada estándar o empleando formularios. En estos lenguajes que iniciaron el mundo de la programación, a las personas les tocaba

hacer uso del ingenio pues todos los procesos debían hacerse desde cero. Hoy en día los lenguajes actuales nos ofrecen muchas ayudas para programar.

El lenguaje GW_BASIC es un entorno de programación clásico. Estos lenguajes evolucionaron en el tiempo, tanto en software como en hardware y se estructuraron en un entorno IDE (integrated development environment), con una estructura de instrucciones mejorada. Este lenguaje se convirtió en Quickbasic y salieron varias versiones (7). Veremos su entorno en la siguiente imagen.

Imagen 3 Entorno QuickBasic 64



Fuente:

PROGRAMACIÓN ESTRUCTURADA

La programación estructurada es conocida como el método de desarrollo de programas más viable porque utiliza tres sistemas de control: la secuencia, la selección y la interacción, las cuales se pueden mezclar para crear programas

que apliquen cualquier requerimiento de procesamiento de datos. La programación estructurada se reconoce como top-down, de arriba hacia abajo.

LA VISIÓN CLÁSICA DE LA PROGRAMACIÓN ESTRUCTURADA

La visión clásica de la programación estructurada se refiere al control de ejecución. El control de su ejecución es una de las cuestiones más importantes que hay que tener en cuenta al construir un programa en un lenguaje de alto nivel. La regla general es que las instrucciones se ejecuten sucesivamente una tras otra, pero diversas partes del programa se ejecutan o no dependiendo de que se cumpla alguna condición. Además, hay instrucciones (los bucles) que deben ejecutarse varias veces, ya sea en número fijo o hasta que se cumpla una condición determinada.

A finales de los años sesenta, surgió una nueva forma de programar que reduce a la mínima expresión el uso de la instrucción goto y la sustituye por otras más comprensibles.

Esta forma de programar se basa en un famoso teorema, desarrollado por Edsger Dijkstra, que demuestra que todo programa puede escribirse utilizando únicamente las tres estructuras básicas de control siguientes:

- Secuencia: el bloque secuencial de instrucciones, instrucciones ejecutadas sucesivamente, una detrás de otra.
- Selección: la instrucción condicional con doble alternativa, de la forma "if condición then instrucción-1 else instrucción-2".
- Iteración: el bucle condicional "while condición do instrucción", que ejecuta la instrucción repetidamente mientras la condición se cumpla.

Los programas que utilizan sólo estas tres instrucciones de control básicas o sus variantes (como los bucles for, repeat o la instrucción condicional switch-case), pero no la instrucción goto, se llaman estructurados.

Ésta es la noción clásica de lo que se entiende por programación estructurada (llamada también programación sin goto) que hasta la aparición de la programación orientada a objetos se convirtió en la forma de programar más extendida. Esta última se enfoca hacia la reducción de la cantidad de estructuras de control para reemplazarlas por otros elementos que hacen uso del concepto de polimorfismo.

LA VISIÓN MODERNA DE LA PROGRAMACIÓN ESTRUCTURADA

La realización de un programa sin seguir una técnica de programación produce frecuentemente un conjunto enorme de sentencias cuya ejecución es compleja de seguir y de entender, pudiendo hacer casi imposible la depuración de errores y la introducción de mejoras. Se puede incluso llegar al caso de tener que abandonar el código preexistente porque resulte más fácil empezar de nuevo.

Cuando en la actualidad se habla de programación estructurada, nos referimos a la división de un programa en partes más manejables (usualmente denominadas segmentos o módulos). Una regla práctica para lograr este propósito es establecer que cada segmento del programa no exceda, en longitud, de una página de codificación, o sea, alrededor de 50 líneas.

Así, la visión moderna de un programa estructurado es un compuesto de segmentos, los cuales puedan estar constituidos por unas pocas instrucciones o por una página o más de código. Cada segmento tiene solamente una entrada y una salida, asumiendo que no poseen bucles infinitos y no tienen instrucciones que jamás se ejecuten.

Cada una de estas partes englobará funciones y datos íntimamente relacionados semántica o funcionalmente. En una correcta partición del programa deberá resultar fácil e intuitivo comprender lo que debe hacer cada módulo.

En una segmentación bien realizada, la comunicación entre segmentos se lleva a cabo de una manera cuidadosamente controlada. Así, una correcta partición del problema producirá una nula o casi nula dependencia entre los módulos, pudiéndose entonces trabajar con cada uno de estos módulos de forma independiente. Este hecho garantizará que los cambios que se efectúen a una parte del programa, durante la programación original o su mantenimiento, no afecten al resto del programa que no ha sufrido cambios.

PROGRAMACIÓN ESTRUCTURADA Y ORIENTACIÓN A OBJETOS

Debido a que una forma habitual de realizar la partición en módulos consiste en agrupar las funciones que permiten operar con un determinado conjunto (o estructura) de datos, existe una analogía evidente con la programación orientada a objetos.

De hecho, ésta puede verse como la continuación lógica de aquélla. Así pues, realizaremos una comparación entre ambas técnicas de programación, resaltando las semejanzas existentes entre ellas.

La mente humana utiliza patrones o modelos que faciliten la comprensión del complejo mundo real. Así, estos patrones mentales de comportamiento abstraen aquellas características de un elemento o situación necesarias para su comprensión ignorando los detalles que resulten irrelevantes. Este proceso es conocido como abstracción y resulta esencial en el razonamiento humano, ya que gracias a él podemos manejar sistemas de gran complejidad.

La abstracción ha sido un aspecto clave de la programación desde sus comienzos. Al principio, los programadores enviaban instrucciones binarias a los ordenadores. Posteriormente, se empleó el lenguaje ensamblador, que no es más que el resultado de un proceso de abstracción del código máquina y que nos permite representar secuencias de bits por símbolos más inteligibles e intuitivos denominados nemotécnicos.

El siguiente nivel de abstracción se alcanzó cuando se crearon las instrucciones definidas por el usuario, las macroinstrucciones que abstraían a las propias instrucciones. En un proceso de mayor abstracción surgieron los lenguajes de alto nivel (como Fortran, C, Pascal, Cobol, etc.), que evitan al programador tener que pelearse con el tan denostado código máquina.

La programación estructurada y la orientación a objetos constituyen los pasos finales de este camino, ya que permiten a los programadores escribir código sin necesidad de conocer los detalles de la implementación de funciones de una librería y objetos respectivamente. Su implementación permanece oculta a los programadores, liberándoles de los condicionantes que esto imponía.

De ese modo, un programador no necesita saber cómo se representan internamente los datos de un tablero de ajedrez. Tan sólo necesitará saber cómo mover, iniciar una partida o validar un determinado movimiento.

Características de la programación estructurada

• Mejor comprensión del código fuente.
• Es más fácil la verificación del código.
• Minimiza el tiempo de prueba.
• Filtración de programas.
• Magnífica presentación.

Conceptos básicos sobre la programación estructurada

- Sistemas de control: es la forma de ejecución de directrices introducidas en ellas, que dependen de requisitos determinados con anterioridad.

- Secuencia: es el orden en que se emplean las instrucciones tal y como se encuentran en el programa, estas instrucciones o bloques señalan la complejidad o tamaño.

- Selección: elección entre dos instrucciones, donde la decisión se toma con base en el examen de un enunciado llamado condición: según se cumpla cada condición se pasa a la siguiente y así sucesivamente.

- Iteración: es una estructura cíclica que se usa para repetir una o más indicaciones al cumplir una condición: si esta se cumple avanza y si no, se regresa al punto donde no se ha dado.

- Fraccionamiento: El fraccionamiento se incorpora para no sobrepasar una página de codificación (50 líneas en promedio), que cumple con propiedades fundamentales:

Las fracciones deben estar relacionadas entre sí, conservando la jerarquía en estructura de árbol.

Las relaciones existentes entre las funciones se deben exhibir claramente para poder comprender las acciones que debe realizar el programa fácilmente.

La comunicación de las fracciones se hace por medio de una lista de parámetros para evitar que se relacionen inadecuadamente.

- Sangrado: Es una técnica que se utiliza para organizar el código por lotes, donde se organizan las instrucciones aplicando espacio o sangría para que sea más fácil su interpretación.

EVOLUCIÓN DE LOS MODELOS Y LENGUAJES DE PROGRAMACIÓN

Los lenguajes de programación fueron creados durante la década de los cincuenta, tras un problema de comunicación entre hombre-máquina ya que el primero usa su idioma natural y el segundo uno artificial, el lenguaje de programación tiene la finalidad de actuar como interfaz entre estos dos para

permitirle al primero controlar la máquina. Entre sus principales propiedades están la exactitud, la eficiencia, la claridad y la portabilidad.

La evolución de los lenguajes de programación hace que estos se clasifiquen en cinco tipos de generaciones. Día tras día son más fáciles de comprender.

Generaciones

• 1GL primera generación: lenguaje máquina estimado de bajo nivel ya que emplea símbolos binarios 1 y 0, es decir son establecidos para cada microprocesador, esta generación puede ser leída por cualquier plataforma.

• 2GL segunda generación: lenguaje simbólico que simplifica la escritura y las instrucciones haciéndolas más legibles, también nombrados ensambladores ya que utilizan códigos.

• 3GL tercera generación: lenguaje de alto nivel que posee comandos adyacentes al lenguaje natural (humano) o matemático. Destaca la facilidad para ser empleado por diferentes fabricantes, es independiente del hardware.

• 4GL cuarta generación: orientada a objetos, contiene un software más avanzado, posee capacidades gráficas, de consulta y bases de datos y de creación de códigos, también admite crear aplicaciones.

• 5GL quinta generación: lenguaje de inteligencia artificial y sistemas expertos que posee muchas características de las cuales se destacan el aumento en la capacidad de memoria, multiprocesadores, más velocidad, formas nuevas de E/S y su capacidad para el proceso de conocimiento.

APLICACIÓN DE LA PROGRAMACIÓN ESTRUCTURADA

Actualmente en el mercado se encuentra gran variedad de aplicaciones basadas en este patrón de programación, muchas de las cuales se encuentran en el comercio, en la banca y en diferentes aplicaciones científicas.

Algunos expertos como Bill Hinshaw, quien está al frente de una empresa que brinda soporte a productos desarrollados en Cobol, afirma que se necesita invertir demasiado tiempo, dinero y recurso humano para cambiar estos programas a nuevas ideas de programación.

Es importante resaltar que actualmente también existen muchas aplicaciones científicas y de análisis numérico desarrolladas en Fortran, (Formula Translator) que nació en 1950 y es uno de los lenguajes de programación orientado a procedimientos.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

Existen diferentes tipos de lenguajes de programación según su operatividad o campo de aplicación, entre estos tenemos:

Según su operatividad

Lenguajes procedimentales: Son lenguajes compilados que utilizan una función encargada de traducir el código fuente a lenguaje máquina para que esta ejecute los comandos o acciones dadas. Generalmente los lenguajes de alto nivel son procedimentales, entre estos tenemos: Basic, Pascal, Fortran, Modula-2, ADA, etc.

Lenguajes declarativos: Para desarrollar programas en estos lenguajes, se utilizan sentencias que permiten describir el problema y determinar la solución, pero no las órdenes requeridas para solucionarlo, por eso se deben declarar o especificar una serie de procedimientos que indican la forma de desarrollar las operaciones utilizando proposiciones, condiciones, ecuaciones, restricciones y afirmaciones que conllevan a la resolución del problema.

Orientados a objetos: Estos lenguajes utilizan la abstracción, que es, identificar las características propias de un objeto para poder realizar operaciones. Los objetos se encapsulan, es decir los datos se almacenan en

variables y las operaciones se ejecutan sobre estas indicando su comportamiento. Los lenguajes más conocidos son: Java; C++, Smalltalk, C# y Python.

Orientados al problema 4gl: Direccionados a problemas específicos, especialmente de gestión, están formados por sentencias que clasifican lo que se pretende realizar. Estos lenguajes admiten la automatización de las actividades de desarrollo de software.

Según su área de desempeño o campo de aplicación

Aplicaciones científicas: Aplicaciones donde prevalecen ejercicios matemáticos o matriciales pertenecientes a algoritmos numéricos. Para estas aplicaciones se encuentran Fortan y Pascal como lenguajes aptos.

Aplicaciones para procesamiento de datos: Usualmente se encuentran ejercicios de mantenimiento, innovación y consulta sobre bases de datos. Se incluyen aplicaciones de gestión empresarial, software para recurso humano, procesos de facturación, software contable, control académico, etc. Para estas aplicaciones se encuentran Cobol y SQL como lenguajes calificados.

Aplicaciones para inteligencia artificial: Estas aplicaciones apuntan a que las máquinas piensen por sí mismas, son preponderantes en este tipo de aplicaciones los sistemas expertos, la robótica, los juegos y la traducción de lenguajes, entre otros. Dentro de estas aplicaciones se incluyen redes neuronales y algoritmos genéticos. Para estas aplicaciones se encuentran los lenguajes LISP y Prolog.

Aplicaciones de tratamiento de textos: Como su nombre lo indica están ligadas al manejo de textos en lenguaje común; C es el lenguaje apropiado para este tipo de aplicaciones.

Aplicaciones para programación de sistemas: La programación de software

de interfaz usuario/hardware está contenida en este tipo de aplicaciones, que funcionan como patrones del sistema operativo y los traductores. Se empleaba usualmente el ensamblador, pero hoy día C, ADA y Modula-2 son los lenguajes calificados para este tipo de aplicaciones.

Lenguajes para internet: Se encuentran muchos lenguajes de internet los cuales emergen de las tendencias y necesidades de las plataformas, entre estos encontramos:

- Lenguajes de marcado: por medio de marcas o etiquetas muestra las características de un documento en texto, entre estos encontramos. Html, xhtml y XML (metalenguaje que establece normas que otros lenguajes de marcado deben cumplir). Este tipo de lenguajes no son de programación ya que no admiten ejecutar ninguna operación y condicionan la información.
- Lenguajes de estilo: reconocido como hojas de estilo en cascada las cuales señalan la manera como han de visualizarse las páginas; solo se encuentra como este tipo de lenguaje el CSS el cual no es considerado como lenguaje de programación.
- Lenguajes de programación del lado del cliente: son efectuados por el navegador en la computadora cliente, es concreto, pequeño y brinda la facultad para desarrollar programas que hacen interactivas o con alguna característica propia a las páginas, entre ellos están: AJAX (aleación entre XML y JavaScript asíncrono), JavaScript y VBScript.
- Lenguajes de programación del lado del servidor: son lenguajes multipropósito, potentes y se ejecutan en el servidor, se emplean para programas de gran tamaño, entre ellos están: ASP.NET, PHP, Perl, Python, Cold Fusion, Ruby on Rails, JSP y Java.
- Gestores de bases de datos: admite el acceso y la manipulación de datos de una BD, se utilizan como intercomunicadores entre el almacén de

datos y un software o producto. Los más empleados son MySQL, Oracle, MS-SQL y PostgreSQL.

CONCLUSIONES Y RECOMENDACIONES

- Se puede asegurar que la programación desde sus inicios conserva la estructura lógica que propició el surgimiento de paradigmas que mejoran los procesos e innovación de nuevos modelos, técnicas y herramientas para el desarrollo de programas, por eso es importante conocer cómo nace, crece, se desarrolla y sigue en constante evolución para mejorar la reusabilidad del código y mejorar la métrica en los desarrollos de sistemas de información.

- La programación estructurada significo un avance enorme para la nueva era de la programación moderna. Ayuda en gran parte al orden y compresión del código fuente, permitiendo con esto mejor entendimiento, mejor control y sobre todo reducción de fallas.

- La POO es el paradigma a la que todos los lenguajes han llegado en el desarrollo de sus plataformas que facilita una sintaxis limpia para los conceptos de clase, herencia, encapsulamiento y polimorfismo, por mencionar algunos de ellos.

REFERENCIAS BIBLIOGRÁFICAS

Eckel, E. (2013). Manual piensa Java. Recuperado de https://www. bibliadelprogramador.com/2013/09/manual-piensa-java-bruceeckel- 4ta.html

Deitel, P. J., y Deitel H. M. (2008). Java. Cómo Programar. Recuperado de https://docs.google.com/file/ d/0B1znxPN5ACgSbHhVU1RaVFh0cVU/edit

Pastor, J. Los lenguajes perdidos: COBOL, Delphi o FORTRAN siguen siendo críticos, pero no hay quien programe en ellos. Recuperado de https://www.xataka.com/aplicaciones/los-lenguajes-perdidoscobol- delphi-o-fortran-siguen-siendo-criticos-pero-no-hay-quienprograme- en-ellos

(s. a.). (s. f.). Programación estructurada. Recuperado de http:// www1.frm.utn.edu.ar/informatica1/VIANI/PROGRAMACION%20 ESTRUCTURADA/PROGRAMACION%20ESTRUCTURADA.PDF

San Marcos

www.usanmarcos.ac.cr

San José, Costa Rica