Introducción a la Programación

TEMAS

A continuación se presentan los temas base para programar en

Programación Estructurada

Una característica importante en la programación estructurada es que puede ser leído en secuencia, sin perder la continuidad de la tarea que cumple el programa. Para programar con este método es necesario conocer ciertos elementos de la programación que se describen a continuación

1. VARIABLES

Definición

En la programación es necesario guardar los valores ya sean datos de entrada o datos de salida en la memoria de la computadora para pode trabajar con ellos durante la ejecución del programa. Estos valores los vamos a guardar en lo que se le conoce como variables.

Este concepto de variable es usado en matemáticas, lógica, estadística, entre otras disciplinas. En la programación se emplea como el símbolo que representa el valor almacenado en memoria y que puede cambiar de valor durante la ejecución del programa.

A las variables, entonces, se les debe especificar que tipo de valor van a almacenar. A estos tipos de valor se les conoce como tipo de datos y se tratarán en el tema 2 de este documento.

Tipos de variables

Existen dos tipos de variables: las variables globales y las variables locales. Las variables globales se usan en cualquier parte del programa. En cambio las variables locales se usan solamente dentro de las funciones donde se declaran.

Sintaxis: declaración y llamada

La sintaxis es la forma general de escribir los elementos de programación, tiene una parte que no va a cambiar y otra que varia dependiendo de la situación, esta última es definida entre los símbolos de menor y mayor que.

En el caso de las variables es necesario indicarle a la computadora que variables se van a usar y de qué tipo de dato van a almacenar antes de usarla a esto se le llama: declaración.

Sintaxis de la declaración de una variable:

<TipoDato> <identificador>**;**

Ejemplo:

int numero;

El tipo de dato se describe en el tema 2 y el identificador se describe en el tema 3 de este documento. La variable nos permitida identificar el valor que almacena en ella.

Después de declarar la variable, entonces podemos hacer uso de ella asignándole el valor que deseamos que tenga.

Sintaxis de una variable para el caso de una asignación por el usuario en el lenguaje de c:

scanf (“<formato>”**,**<valor inicial>**);**

Sintaxis de una variable para el caso de asignarle un valor inicial:

<identificador> **=** <valor inicial>**;**

Sintaxis de la llamada de una variable para el caso de asignarle una operación:

<identificador> **=** <operacion>**;**

AUTOR
PROF. SABRINA LIZBETH VEGA MALDONADO

PROGRAMACIÓN

CONCEPTOS BÁSE

A continuación se definen conceptos que son base para iniciar con el aprendizaje de la programación.

- Programación

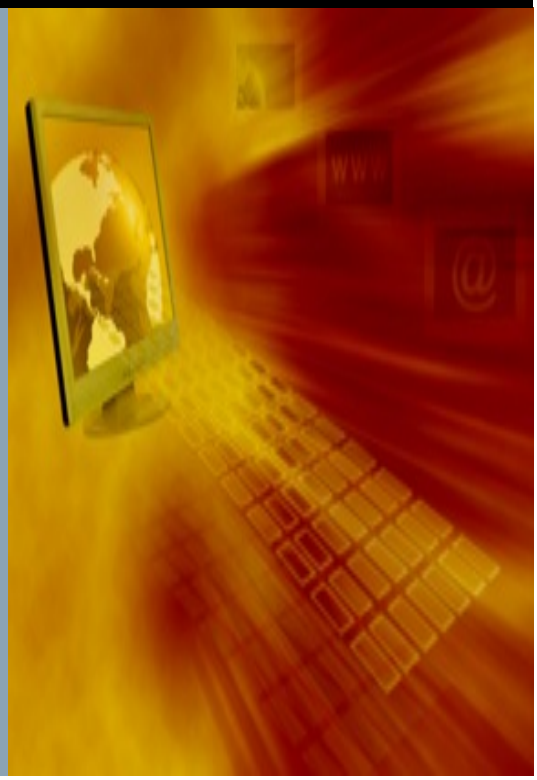
Es la técnica para dar instrucciones a la computadora para que solucione un problema determinado. Existen diferentes métodos para la programación, uno de ellos es la programación estructurada

- Programa

Son un conjunto de instrucciones definidas por un lenguaje de programación

- Lenguaje de programación

Al igual que un lenguaje humano, los lenguajes de programación tienen ciertas sintaxis y gramática que debemos implementar en las instrucciones



En el lenguaje de C es necesario indicar el formato del tipo de dato que se está usando en la llamada de la variable. Este formato depende del valor almacenado, es decir, del tipo de dato. A continuación se enlistan:

%d	Número entero positivo o negativo
%c	Un carácter (letra o símbolo)
%s	Una cadena
%f	Un número real positivo o negativo
%e	Un número en notación científica

Ejemplos

Supongamos que tenemos ya diseñados los dato de entrada numero que se le pedirá al usuario y el dato de salida cuadrado donde se guardará el proceso de elevar el número dado al cuadrado.

La declaración de las variables siguiendo la sintaxis y la clasificación de tipos de datos del tema 2 será:

```
int  numero;
int cuadrado;
```

La llamada de las variables siguiendo la sintaxis será:

```
printf("Escribe un numero entero: ");
scanf("%d", numero);
cuadrado = numero * numero;
printf("El numero elevado al cuadrado es: %d", cuadrado);
```

2. TIPO DE DATOS

Definición

Los tipos de datos como ya se abordó en el tema anterior dependen del valor que se quiera usar, es decir, es la clasificación de los valores que pueden ser usados en la programación.

Clasificación

En el lenguaje de C los tipos de datos se clasifican de la siguiente manera:

Tipo de Dato	Requisito de Almacenamiento	Rango
int	4 bytes (con signo)	-2,147,483,648 a 2,147,483,647
short	2 bytes (con signo)	-32,768 a 32, 767
long	8 bytes (con signo)	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
byte	1 byte (con signo)	-128 a 127
float	4 bytes (con signo)	3.4 e-308 a 3.4e+308 (6-7 cifras decimales significativas)
	8 bytes (con signo)	1.7 e -308 a 1.7 e +308 (15 cifras decimales significativas)
char	2 bytes (sin signo)	\u0000 a \uFFFF (caracter Unicode)

Fuente: Luis Aguilar Joyanes "Programación C, C++, Java, UML"

3. IDENTIFICADORES

Definición

Los identificadores es la forma en como vamos a nombrar a los diferentes elementos de la programación. Los identificadores que establece el programador son usados principalmente en las variables y funciones, por lo que deben tener un nombre significativo con el valor de la variable o proceso de la función.

Entonces para elegir el identificador de por ejemplo una variable, dependerá del valor que se guarde en ella; tal es el caso para una variable donde se requiere guardar como valor un número telefónico, sabemos entonces que aunque podemos nombrarla con el identificador que sea, el identificador más apropiado será: telefono.

Los identificadores siguen características que están establecidas y se pueden definir como reglas para establecer un identificador, estas características se describen a continuación.

Características

Las características de un identificador se usan como reglas estrictas dentro del lenguaje de programación y se describen de la siguiente manera:

- puede inicia con una letra o un guión bajo
- consta de uno o más caracteres
- su longitud será de máximo 31 caracteres
- se distingue entre mayúsculas y minúsculas
- pueden emplearse dos o más palabras juntas o separadas con un guión (medio o bajo)
- no pueden tener espacios en blanco
- no pueden existir dos identificadores iguales
- no puede iniciar con un símbolo o con un número
- no puede tener espacio en blanco

4. OPERADORES

Los operadores en la programación son elementos que se usan en ciertos cálculos Estos operadores se clasifican en tres: operadores aritméticos, operadores relacionales y operadores lógicos.

Operadores aritméticos

Los operadores aritméticos son usados para crear expresiones matemáticas que realizan cálculos u operaciones tales como suma, resta, multiplicación división, módulo, raíz cuadrada, potencia, entre otras.

En el lenguaje de C son utilizados de la siguiente manera:

Operador aritmético	Operación	Ejemplo	Resultado
+	Suma	x=4+3;	x=7
-	Resta	x=4.5-3;	x=1
*	Multiplicación	x=4.5*3;	x=12
/	División	x=4/3; v=(float) 4/3;	x=1 v=1.33
%	Módulo (residuo)	x= 15%2; v=((float) (15%2))/2;	x=1; v=0.5
++	Incremento	y=x++; y=++x;	y=7; y=8;
--	Decremento	y=x--; y=--x;	y=6; y=5;

Fuente: Osvaldo Cairo “Fundamentos de Programación. Piensa en C”

Operadores relacionales

Los operadores relacionales son símbolos que se usan para comparar dos valores.

En el lenguaje de C son utilizados de la siguiente manera:

Operador relacional	Operación	Ejemplo	Resultado
==	Igual a	res='h' == 'p'	res=0
!=	Diferente de	res='a' != 'b';	res = 1
<	Menor que	res= 7 < 15;	res=1
>	Mayor que	res = 22 > 11;	res=1
<=	Menor o igual que	res = 15<=2;	res=0
>=	Mayor o igual que	res = 35>=35;	res=1

Fuente: Osvaldo Cairo "Fundamentos de Programación. Piensa en C"

Operadores lógicos

Los operadores lógicos producen un resultado booleano, es decir, comprueban la veracidad o falsedad de una expresión condicional.

En el lenguaje de C son utilizados de la siguiente manera:

Operador lógico	Operación	Ejemplo	Resultado
!	Negación	x = (!(7<15)); /*(!0) ->1* y = (!0);	x=1 y=1
&&	Conjunción	x = (35>20)&&(20<=23); /* 1 && 1 */ y = 0 && 1;	x=1 y=0
||	Disyunción	x = (35>20)||(20<=18); /* 1 || 0*/ y = 0 || 1;	x=1 y=1

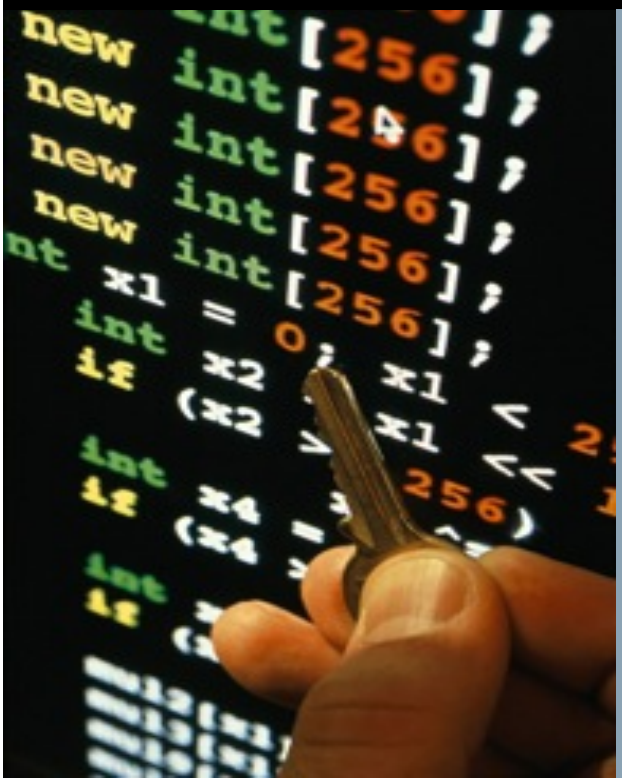
Fuente: Osvaldo Cairo "Fundamentos de Programación. Piensa en C"

PALABRAS RESERVADAS

EN EL LENGUAJE DE C

auto	continue	enum	if	short	switch
break	default	extern	int	signed	typedef
case	do	float	long	sizeof	union
char	double	for	register	static	unsigned
const	else	goto	return	struct	void

Fuente: Osvaldo Cairo "Fundamentos de Programación. Piensa en C"



ESTRUCTURA DE UN PROGRAMA

EN EL LENGUAJE DE C

La estructura de un programa en el lenguaje de C tiene los siguientes elementos:

- Cabecera y/o directivas
- Declaración de variables globales
- Definición de funciones

El ejemplo de un programa que imprime el valor de una variable siguiendo la estructura es el siguiente:

```
#include <stdio.h>
int numero;
void main()
{
  numero = 125;
  printf("El valor del numero es: %d", numero);
}
```

5. ARREGLOS

Definición

Los arreglos son una colección de variables del mismo tipo que se referencian utilizando un nombre común. Un arreglo consta de posiciones de memoria contigua. La dirección más baja corresponde al primer elemento y la más alta al último. Un arreglo puede tener una o varias dimensiones. Para acceder a un elemento en particular de un arreglo se usa un índice.

Los arreglos de una dimensión se les conoce como arreglos unidimensionales y los arreglos de varias dimensiones se les conoce como arreglos multidimensionales

Sintaxis arreglo unidimensional

La sintaxis para declarar un arreglo unidimensional en el lenguaje de C es la siguiente:

<tipoDato> <identificador> **[** <longitud> **];**

La declaración del arreglo en el lenguaje de C puede hacerse en el área de declaración de variables globales y podrá ser usado en cualquier parte del programa o bien puede declararse dentro de una función haciendo al arreglo local y sólo será usado en dicha función.

Por ejemplo, podemos tener un arreglo que almacene 20 calificaciones, su declaración quedaría:

float calificaciones [20];

5. FUNCIONES

Definición

Las funciones son procedimientos o subrutinas que se utilizan para facilitar la solución del problema.

Las funciones se dividen en dos categorías:

- Funciones que regresan valor
- Funciones que no regresan valor

El paso de parámetros en una función es opcional, esto quiere decir que si la fundión necesita o va a trabajar con valores que no se encuentran dentro de ella se le puede declarar parámetros para que en ellos se asignen esos valores de afuera. Los parámetros entonces sirven para comunicar a una función con otra.

Las funciones se tienen primero que definir y después se hace una llamada de ella para que se ejecute. Su sintaxis es la siguiente

Sintaxis

Funciones que regresan valor:

- Sintaxis de la definición:

<tipoDato> <identificador> ([<parametros>])
{
<sentencia 1>**;**
…
<sentencia n>**;**
}

- Sintaxis de la llamada:

<variable>=<identificador> ([<argumentos>]);

Nota1: Una vez que la llamada de una función que regresa valor se asigna a una variable (declarada con anticipación) es importante imprimir en pantalla esa variable para que el usuario pueda ver el resultado.

Funciones que no regresan valor:

- Sintaxis de la definición:

void <identificador> ([<parametros>])
{
<sentencia 1>**;**
…
<sentencia n>**;**
}

- Sintaxis de la llamada:

<identificador> ([<argumentos>]);

Nota 2: La llamada de una función ya sea que regresen valor o que no regresen valor se debe programar en otra función.

Ejemplos en el lenguaje C

Funciones que regresan valor:

```
//cabecera
#include <conio.h>
#include <stdio.h>

//para este ejemplo no hay variables globales

//definición de funciones
int suma (int a, int b)
{
      return (a+b);
}
void main( )  //función principal
{
      //declaración variables locales
      int valor1, valor2, resultado;
      //datos de entrada
      printf("Escribe el primer valor : ");
      scanf("%d",valor1);
      printf("Escribe el segundo valor: ");
      scanf("%d",valor2);
      //proceso
      resultado=suma(valor1, valor2); //llamada de función
      //datos de salida
      printf("La suma es=%d",resultado);
      getch( );
}
```

Funciones que no regresan valor:

```
//cabecera
#include <conio.h>
#include <stdio.h>

//variables globales
int resultado;

//definición de funciones
void suma (int a, int b)
{
      resultado = a+b;
      printf("La suma es=%d",resultado); //dato de salida
}
void main ( )  //función principal
{
      //declaración variables locales
      int valor1, valor2;
      //datos de entrada
      printf("Escribe el primer valor : ");
      scanf("%d",valor1);
      printf("Escribe el segundo valor: ");
      scanf("%d",valor2);
      //proceso
      suma(valor1, valor2); //llamada de función
      getch( );
}
```

Nota 3: Una función que regresa valor o que no regresa valor también puede tener o no parámetros esto depende de si se declaran las variables globales o locales:

Si las variables se declaran globales, pueden ser usadas en cualquier parte del programa entonces las funciones no necesitaran de parámetros.

Si las variables se declaran locales, sólo las puede ver la función donde se declararon por lo tanto si esos valores se necesitan en otra función, ésta necesitará de paso de parámetros.