Fundamentos para la construcción de código a partir del algoritmo

Objetivo: El alumno construirá programas utilizando el lenguaje de programación **C** a través de un análisis y modelado algorítmico previo.

Contenido:

3.1. Sintaxis básica y semántica.
3.2. Variables, tipos, expresiones y asignación.
3.3. Estructuras de control condicional e iterativo.
3.4. Funciones y paso de parámetros.
3.5. Descomposición estructurada.
3.6. Manejo de E/S.
3.7. Estrategias de depuración.
  3.7.1. Tipo de errores.
  3.7.2. Técnicas de depuración.

CICLO DE VIDA DE UN PROGRAMA

Al igual que en la resolución de problemas, existen ciertos pasos que debemos seguir para la creación de programas, estos son:

Análisis del problema

Elaborar el Algoritmo

Codificación del algoritmo

Depuración de código

Mantenimiento de programa

Codificación del algoritmo

Después de tener el algoritmo, el siguiente paso es codificarlo en el lenguaje que seleccionamos, para este caso en específico es **Lenguaje "C".**

Programación Estructurada

Algoritmo → **Programa→** Traducción del Algoritmo a un Lenguaje de Programación.

Programa → Estructuras de Datos, Operaciones Primitivas y Estructuras de Control.

Estructura → Representación concreta del estado de una entidad.

Estructura de Datos → Métodos que se emplean en programación para organizar y representar la información en una computadora.

Estructuras de Control → La estructura de un programa, definida por referencias con las cuales se representan las transferencias de control. Construcciones mediante las que se escriben los programas.

Operaciones Primitivas → Acciones que se ejecutan sobre los datos para transformarlos en información.

Lenguaje C

Creado entre 1970 y 1972 por Brian Kernighan y Dennis Ritchie para escribir el código del sistema operativo UNIX.

Desde su nacimiento se fue implantando como el lenguaje de programación de sistemas, sobre todo por ser un lenguaje que conjugaba la abstracción de los lenguajes de alto nivel con la eficiencia del lenguaje máquina.

A mediados de los ochenta el C se convierte en un estándar internacional ISO. Este estándar incluye tanto la definición del lenguaje como una enorme biblioteca de funciones para entrada/salida, tratamiento de textos, matemáticas, etc.

Los programas en Lenguaje C se escriben en un editor de texto cualquiera, por ejemplo vi en Linux, bloc de notas en Windows.

Una vez escrito el programa, este debe compilarse, es decir, debe ser leído por un programa llamado

"**compilador**" que lo traduce a lenguaje de máquina y produce un nuevo archivo que enlaza con las bibliotecas para poder generar el programa ejecutable.

La manera estándar de compilar un programa en Lenguaje C en Linux es abrir una terminal, ir al directorio que contiene el programa y escribir:

gcc programa1.c

Al compilar el programa se obtiene un ejecutable que tiene por default el nombre:

a.out

Esto puede cambiarse haciendo:

gcc programa1.c -o programa1

La opción -o le dice al compilador que el ejecutable debe llamarse "***programa1***". Para ejecutar el programa se escribe simplemente:

./programa1

donde se usa **./** para indicar a Linux que debe buscar el ejecutable en el directorio local.

Investigar –wall para el proceso de compilar

Características del Lenguaje C

Alfabeto o conjunto de caracteres:

Caracteres alfabéticos

Caracteres numéricos

Caracteres especiales

Vocabulario o léxico

Conjunto de palabras válidas o reservadas en el Lenguaje C.

Gramática

Conjunto de lineamientos que se deben respetar para lograr construir frases, oraciones o instrucciones, se logra transmitir a la computadora que se deseamos hacer.

Características de un buen programa

Operatividad → Funcional.

Legibilidad → Algo de espacios o sangrías, para reflejar las estructuras de control.

Transportabilidad → Se puede ejecutar sin hacer modificaciones.

Claridad → La comunicación de lo que desea realizar, buenos identificadores e incluso comentarios dentro del programa.

Modularidad → Dividir el programa puede contribuir a realizar las tareas de manera mas clara, además que permite reutilizar el código.

Componentes del Lenguaje C

Declaraciones. Su objetivo es dar a la computadora la información sobre tipos de variables, arreglos y características diversas, en caso de C, también incluye la posibilidad de dar valores iniciales a las variables.

Instrucciones Ejecutables. Son aquellas en las que se calcula o realiza algo, toda instrucción que implica un cambio.

Esquema de un Programa Fuente

Directivas
Declaraciones Globales
Función Principal
Funciones Secundarias

#include <biblioteca1.h>
#include <biblioteca2.h>
... declaraciones de funciones...
... definiciones (cuerpos de funciones)...
... declaraciones de variables globales...

main() /* nombre de la función principal*/
{
... cuerpo del main...
}
...otras definiciones de funciones...

Esquema de la Función "main"

main()
{
…declaraciones de variables locales…
…instrucciones a ejecutar…
}

Directivas más comunes

#include <librería.h>
#define expresión

Archivos de cabecera mas utilizadas

Indican que usaremos funciones de tipo:
Entrada y salida de datos (stdio.h)
Rutinas matemáticas (math.h)
Manejo de cadenas (string.h)

Estructura de un programa en Lenguaje C con comentarios.

Ejem1.

```
#include<stdio.h>
    /*Programa. Uso de comentarios*/

int main(){
    //Programa de prueba
    return 0;
}
// Nombre del archivo: prueba1.c
// Compilado: gcc prueba1.c –o prueba1
//Ejecutado: ./prueba1
```

Salida de datos –Texto-

printf("Texto que se muestra en pantalla"**);**

Ejem2.

```
#include<stdio.h>
int main(){
    // Este programa es una prueba para mostrar texto
     printf("\nSaludos al grupo de Fundamentos de Programación\n");
    return 0;
}
// Nombre del archivo: prueba2.c
// Compilado: gcc prueba2.c –o prueba2
//Ejecutado: ./prueba2
```

Tipos de Datos

En C se disponen de estos tipos básicos:

int	enteros (números enteros positivos y negativos)	4 byte → 32 bits
char	caracteres (letras)	1 byte → 8 bits
float	números en coma flotante (números reales)	4 byte → 32 bits
double	números en coma flotante de doble precisión	8 byte → 64 bits

Tipos modificados

Modificador	Significado
short	entero corto (rango más pequeño)
long	entero largo (rango más amplio)
unsigned	entero sin signo (0..N)
signed	entero con signo (-N-1 .. +N)

Declaraciones de variables

Variable: Almacenamiento de DATOS

Hay que declarar las variables antes de usarlas y cada variable tiene un tipo.

Ejemplo:

```
int x;
float y, z;
```

Operadores Aritméticos

Los datos se manipulan mediante expresiones, que sirven para calcular valores. En C hay varios operadores para construir expresiones.

Estos son los operadores elementales sobre números:

Operadores aritméticos

OPERADOR	OPERACIÓN
()	Paréntesis
%	Módulo (resto de la división entera)
*	Multiplicación
/	División
+	Suma
-	Resta

Recuerden las jerarquías de los operadores aritméticos.

Una expresión combina varias operaciones y devuelve un valor.

Los operadores *, / y % tienen precedencia sobre la suma y la resta.

Se pueden utilizar paréntesis para agrupar subexpresiones.

Asignaciones

La forma de dar valor a una variable es

variableSuma= expresión_aritmética;

También se puede dar valor a una variable en el mismo momento en que se declara (inicialización).

tipo variable = expresión;

Salida de datos –Contenido de Variables-

printf (“cadena de formato”, arg1, arg2, ... argN);

En la cadena de formato aparecen:

- el texto que se desea imprimir
- caracteres especiales
- secuencias de escape
- indicaciones del formato de los argumentos

Los argumentos son expresiones cualesquiera.

Para usar printf, hay que escribir al principio del programa la directiva **#include <stdio.h>**

Formatos de printf

%d	%i	Número Entero
%c	%s	Carácter, cadena
%f		Número Real

Secuencias de escape

\n	Salto de línea
\t	Tabulación
\a	Sonido

Ejem3.

```
/*bibliotecas*/
#include<stdio.h>
//declaración de variables globales
int main(){
    //declaración de variables locales
    int suma;
    suma=2+9;
    printf("%d\n",suma);
    return 0;
}
    // Nombre del archivo: serie1.c
    // Compilado: gcc serie1.c –o serie1
    //Ejecutado: ./serie1
```

Entrada de datos

scanf (“formato”, & arg1, & arg2, ...);

Sintaxis:

En formato se especifica qué tipo de datos se quieren leer. Se utiliza la misma descripción de formato que en printf. También hay que incluir la cabecera **<stdio.h>**

Ejem4.

```
#include<stdio.h>
int main(){
    float a, b, suma;
    printf("Ingrese a: ");
    scanf("%f",&a);
    printf("Ingrese b: ");
    scanf("%f",&b);
    suma=a+b;
    printf("%f\n",suma);
    return 0;
}
```

// Nombre del archivo: ***serie2.c***
// Compilado: ***gcc serie2.c –o serie2***
//Ejecutado: ***./serie2***