



Repaso de Programación Estructurada

Lenguaje C



UBGVIRTUAL

Algoritmo

Programación Estructurada



+ Un algoritmo es una serie de pasos ordenados, finitos (inicio/fin), precisos y bien definidos que permiten resolver un problema o una tarea específica computable.

+ Características

- + Identificar datos de entrada
- + Identificar proceso(s)
- + Identificar datos de salida
- + Esta representación puede ser escrita a lo que se le conoce como pseudocódigo o bien puede ser gráfica a lo que se le conoce como diagrama de flujo



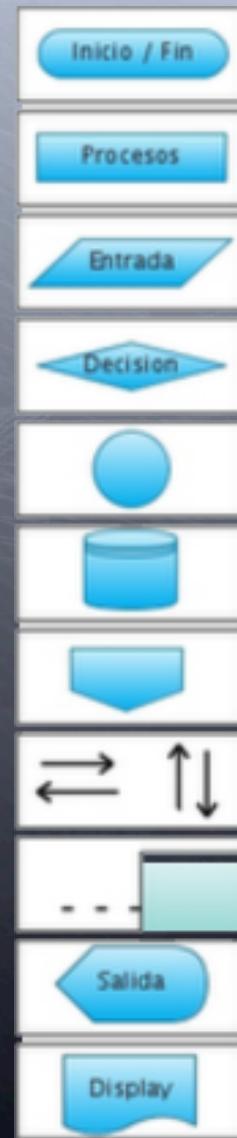
Diagrama de Flujo

Diseño



+ representación del algoritmo a través de símbolos.

+ Simbología:



Inicio o Fin del programa

Pasos, procesos o líneas de instrucciones

Datos de entrada o salida

Toma de decisión y ramificación

Conector para el flujo a otra parte del diagrama

Disco magnético

Conector de página

Líneas de flujo

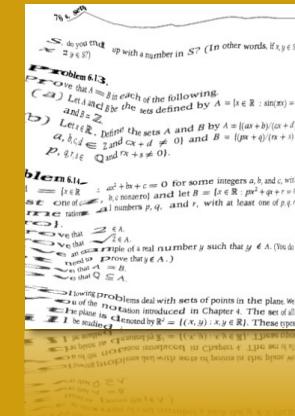
Anotación

Display, para mostrar datos

Mandar datos a la impresora

Pseudocódigo

Diseño



+ Se utiliza como intermediario entre la solución del problema expresado en el lenguaje que las personas podemos comprender y el lenguaje que la computadora puede entender.

+ Ejemplo:

1. Inicio
2. imprimir en pantalla el mensaje al usuario de que escriba un valor entero
3. guardar el primer valor en la variable numero1
4. imprimir en pantalla el mensaje al usuario de que escriba otro valor
5. guardar el segundo valor en la variable numero2
6. asignar la operación a la variable resultado:
`resultado=numero1+numero2`
7. imprimir en pantalla el valor de la variable resultado
8. final

Bases Lenguaje C

Programación Estructurada



Tipos de Datos

+ Tipos de Datos

| 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) |
| double | 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"

Identificadores

Palabras reservadas en el lenguaje 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"

Operadores Aritméticos

| 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 Aritméticos simplificados

| Operador aritmético | Forma simplificada de uso | Ejemplo | Equivalencia |
|---------------------|---------------------------|----------------------|-----------------------|
| + | <code>+=</code> | <code>x+=5;</code> | <code>x=x+5;</code> |
| - | <code>-=</code> | <code>x-=3;</code> | <code>x=x-3;</code> |
| * | <code>*=</code> | <code>x*=4.5;</code> | <code>x=x*4.5;</code> |
| / | <code>/=</code> | <code>x/=3;</code> | <code>X=x/3;</code> |
| % | <code>%=</code> | <code>x%=12;</code> | <code>X=x%12;</code> |

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

⊕ Operadores Relacionales

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

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

⊕ Operadores Lógicos

| 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"

Arreglos

- + Es una variable capaz de guardar de uno a n valores.
Su sintaxis es: <tipoDato> <identificador> [<longitud>];
- + Ejemplo:
 - + Int calificación [100];

0
1
2
. .
99

+ Usando Arreglos

```
#include <conio.h>
#include <stdio.h>
void main() {
    int valores[5]={100, 200, 300, 400, 500};
    printf("El arreglo contiene los siguientes valores:\n");
    for (int i=0; i<5; i++)
        printf("Valor %d : %d", (i+1), valores[i]);
    getch();
}

#include <conio.h>
#include <stdio.h>
void main() {
    int calificacion[5];
    int suma=0;
    for (int j=0; j<5; j++) {
        printf("Escribe la calificacion %d : ", (j+1));
        scanf("%d", &calificacion[j]);
        suma += calificacion[j];
    }
    printf("El promedio es=%d", (suma/5));
    getch();
}
```

Bases de la metodología

Programación Estructurada

Funciones

- + 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



+ Funciones que regresan valor

+ Sintaxis de la definición

```
<tipoDato> <identificador> ( [<parametros>] ) {  
  
    <sentencia 1>;  
  
    ...  
  
    <sentencia n>;  
  
}
```

+ Sintaxis de la llamada

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

+ Funciones que no regresan valor

+ Sintaxis de la definición

```
void <identificador> ( [<parametros>] ) {  
  
    <sentencia 1>;  
  
    ...  
  
    <sentencia n>;  
  
}
```

+ Sintaxis de la llamada

```
<identificador> ( [<argumentos>] );
```

Paso de Parámetros

Los parámetros permiten establecerle a una función valores que se encuentra fuera de ella.

En el lenguaje de C, existen dos tipos de parámetros que son: por valor y por referencia



+ Usando Funciones

```
#include <conio.h>
#include <stdio.h>
Int suma (int a, int b) {
    return (a+b);
}
void mainl () {
    int valor1, valor2, resultado;
    printf("Escribe el primer valor : ");
    scanf("%d",valor1);
    printf("Escribe el segundo valor: ");
    scanf("%d",valor2);
    resultado=suma(valor1, valor2);
    printf("La suma es=%d",resultado);
    getch( );
}
```

Estructuras de Control

Reglas Básicas de Programación

Secuencial

Se coloca una sentencia después de otra.

Una secuencia es el elemento más simple de la programación





```
#include <conio.h>
#include <stdio.h>
void mainl () {

    printf("Hola mundo \n");
    printf("Presiona cualquier tecla para salir...");
    getch();
}

}
```

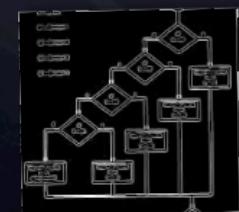
Selectiva

- Selectiva simple

- Selectiva doble

- Selectiva múltiple

- + En esta estructura de control se evalua una condición booleana para realizar un o varias sentencias o bien realizar otras sentencias.
- + Para establecer la condición booleana se utilizan los operadores relacionales



Sintaxis

Selectiva simple en lenguaje C

```
if (<condición>) {  
    <sentencia 1>;  
    ...  
    <sentencia n>;  
}
```

LA SINTAXIS EN PSEUDOCÓDIGO SÓLO SE CAMBIARÍA LAS PALABRAS EN INGLÉS POR ESPAÑOL

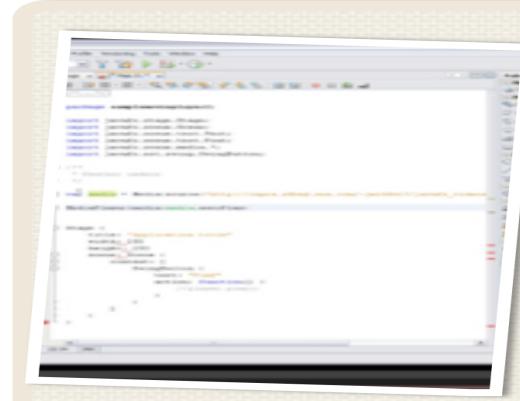


EJEMPLO

```
if (maximo>=50) {  
    System.out.println("fuera de rango");  
    System.out.println(maximo);  
}
```

Selectiva doble en lenguaje C

```
if (<condición>) {  
    <sentencia 1>;  
    ...  
    <sentencia n>;  
}  
  
else {  
    <sentencia 1>;  
    ...  
    <sentencia n>;  
}
```

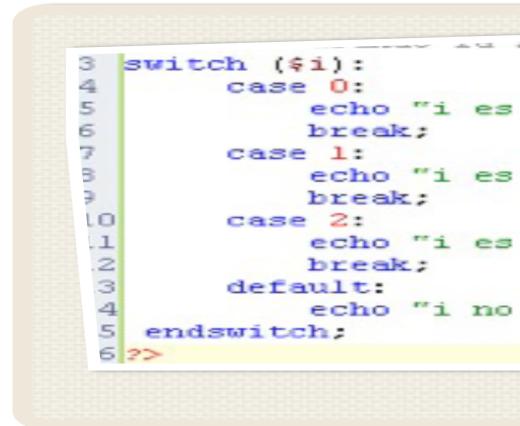


EJEMPLO

```
if(promedio>=60) {  
    System.out.println("APROBADO");  
    System.out.println(promedio);  
}  
else {  
    System.out.println("REPROBADO");  
    System.out.println(promedio);  
}
```

Selectiva múltiple en lenguaje C

```
switch (<expresión>) {  
    case 1:<sentencia 1>;  
    ...  
    <sentencia n>;  
    break;  
    ...  
    case n:<sentencia 1>;  
    ...  
    <sentencia n>;  
    break;  
    [default:<sentencias>:]  
}
```



EJEMPLO

```
switch (opcion) {  
    case 1: resp=a+b;  
    break;  
    case 2: resp=a-b;  
    break;  
    case 3: resp=a*b;  
    break;  
    case 4: resp=a/b;  
    break;  
    case 5: resp=Math.sqrt(a);  
    break;  
    case 6: resp=Math.sqrt(b);  
    break;  
}
```

Ejemplo de seletiva anidadas

```

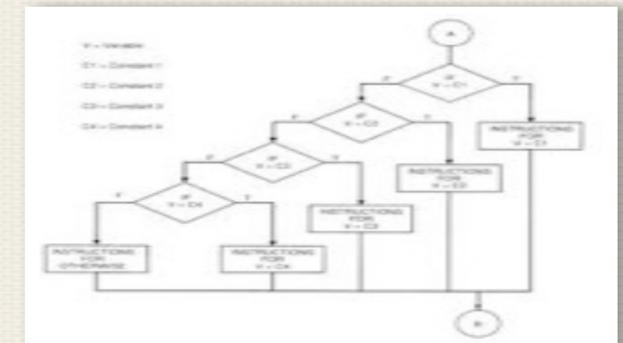
include <conio.h>;
include <stdio.h>;
void main( ) {
    int mes;
    printf("Escribe el mes del año con numero: ");
    scanf("%d", &mes);
    if(mes==1)
        printf( "El mes es Enero \nLa estación es Invierno");
    else
        if(mes==2)
            printf("El mes es Febrero \nLa estación es Invierno");
        else
            if(mes==3)
                printf( "El mes es Marzo\nLa estación es Primavera");
            else
                if(mes==4)
                    printf("El mes es Abril \nLa estación es Primavera");
                else
                    if(mes==5)
                        printf("El mes es Mayo \nLa estación es Primavera");
                    else
                        if(mes==6)
                            printf( "El mes es Junio \nLa estación es Verano");
                        else
                            if(mes==7)
                                printf("El mes es Julio \nLa estación es Verano");
                            else
                                if(mes==8)
                                    printf("El mes es Agosto \nLa estación es Verano");
                                else
                                    if(mes==9)
                                        printf("El mes es Septiembre \nLa estación es Otoño");
                                    else
                                        if(mes==10)
                                            printf("El mes es Octubre \nLa estación es Otoño");
                                        else
                                            if(mes==11)
                                                printf( "El mes es Noviembre \nLa estación es Otoño");
                                            else
                                                if(mes==12)
                                                    printf("El mes es Diciembre \nLa estación es Invierno");
                                                else
                                                    printf( "numero de mes fuera de rango" );

    printf("\n Presiona cualquier tecla para salir...");
    getch( );
}

```

Las estructuras de control anidadas se utilizan intercambiando la sentencia por otra estructura de control, en este ejemplo la sentencia que va en el “else” se cambio por otra estructura de control selectiva doble de esta manera va evaluando la comparación de forma anidada de tal manera que entra a las sentencias de un solo mes.

Se pueden anidar estructuras de control de diferentes tipos, es decir, en una estructura selectiva simple se puede anidar una estructura selectiva doble o viceversa o bien en una estructura de control selectiva múltiple se pueden anidar estructura selectiva simple, etc. Esto dependerá al cien por ciento de planteamiento del problema.



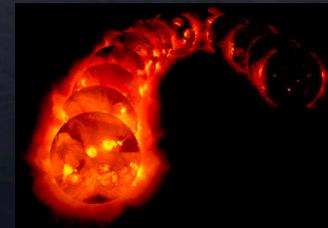
Repetitiva

Ciclo: mientras

Ciclo: hacer mientras

Ciclo: para

- + En estas estructuras de control se repetirán las sentencias que se encuentren dentro de ellas mientras la condición booleana sea verdadera



+ Sintaxis y Ejemplo del ciclo – Mientras –

```
While ( <condición> ) {  
    <sentencia 1>;  
    ...  
    <sentencia n>;  
}
```

```
#include <conio.h>  
#include <stdio.h>  
void mainl () {  
    int tabla, numero;  
    int contador=1;  
    printf("Dame la tabla a multiplicar: ");  
    scanf("%d",tabla);  
    printf("Hasta que numero: ");  
    scanf("%d",numero);  
    while ( contador<=numero) {  
        printf("%d x %d = %d", tabla, contador,(tabla*contador));  
        contador++;  
    }  
    getch();  
}
```

+ Sintaxis y Ejemplo del ciclo – hacer Mientras –

```
do {  
    <sentencia 1>;  
    ...  
    <sentencia n>;  
} While ( <condición> );
```

```
#include <conio.h>  
#include <stdio.h>  
void mainl () {  
    char respuesta;  
    do {  
        printf("Hola mundo \n");  
        printf("Quieres volver a ejecutar? s/n:");  
        scanf("%c", respuesta);  
    } while(respuesta=='s');  
    printf("Presiona cualquier tecla para salir...");  
    getch();  
}
```

+ Sintaxis y Ejemplo del ciclo – Para –

```
for ( <inicia contador>; <condicion>; <incrementa/decrementa> ) {  
    <sentencia 1>;  
    ...  
    <sentencia n>;  
}
```

```
#include <conio.h>  
#include <stdio.h>  
void mainl () {  
    int tabla, numero;  
    printf("Dame la tabla a multiplicar: ");  
    scanf("%d",tabla);  
    printf("Hasta que numero: ");  
    scanf("%d",numero);  
    for ( int i=1; i<=numero; i++) {  
        printf("%d x %d = %d", tabla, i,(tabla*i));  
    }  
    getch();  
}
```