PROGRAMACION EN LENGUAJE C

ALGORITMOS Y ESTRUCTURAS DE DATOS

LIBRERÍAS

Nos aportan funciones elementales para poder resolver las distintas problemáticas.

Se invocan con la sentencia “#include <>”, al inicio del programa.

Las librerías que usamos son:

<stdio.h> : Provee las operaciones más elementales como lectura y escritura.

<conio.h> : Provee funciones únicas para la consola de Windows.

<string.h> : Provee las operaciones necesarias para poder manejar strings.

<stdlib.h> : Provee las operaciones necesarias para poder manejar listas.

TIPOS DE DATOS

- int : El tipo entero, su carácter identificador es “d”.
- char : El tipo carácter, su carácter identificador es “c”.
- float : El tipo flotante o real, su carácter identificador es “.xf”, siendo ‘x’ los decimales.
- Los tipos restantes (booleano y cadena de caracteres) se definen de la siguiente forma:
- Booleano : Se simula con el tipo int, dando el valor 0 para FALSE y cualquier número distinto de 0 para TRUE.
- String : Se forma mediante un arreglo de caracteres. El carácter identificador para esta construcción es “s”.

OPERADORES LÓGICOS

Operador	Símbolo
Igual	==
Distinto	!=
Not	!
Menor	<
Menor o igual	<=
Mayor	>
Mayor o igual	>=
AND	&&
OR	||

DECLARACIÓN DE VARIABLES

Pseudocodigo

```
variableEntera: entero[3]
variableCaracter: carácter[1]
variableFlotante: real[5,2]
variableString: carácter[1+n]
```

Standard C

```
int variableEntera
char variableCaracter
float variableFlotante
char variableString[10]
```

DECLARACIÓN MODULO PRINCIPAL

Pseudocodigo

```
Programa: nombrePrograma
    .
    .
[Declaración de variables]
Hacer
    .
Fin hacer
Fin Programa
```

Standard C

```
[Declaración de variables
globales]

int main(){
  [declaración de variables
locales]
  [bloque de código]
}
```

OPERACIONES LEER Y ESCRIBIR

Pseudocodigo

```
Imprimir(“Ingrese un dato entero: ”)

Leer(valor)

Imprimir(“Ingreso: ”,valor)
```

Standard C

```
printf("Ingrese un dato entero: ");

scanf("%d", &valor);

printf("\nIngreso: $d", valor);
```

ESTRUCTURA DE DECISIÓN

Pseudocodigo

```
Si (a != b) Entonces
  [bloque de código]

Sino
  [bloque de código]
Fin Si
```

Standard C

```
if (a != b){
  [bloque de código]
}
else{
  [bloque de código]
}
```

ESTRUCTURA DE CASO

Pseudocodigo

```
Caso (valor)

    1:
     Imprimir("1")
    2:
     Imprimir(“2”)
    3:
     Imprimir(“3”)
    EnOtroCaso:
     Imprimir("Rama falsa")

Fin Caso
```

Standard C

```
switch (valor){
  case 1:
    printf("1");
    break;
  case 2:
    printf("2");
    break;
  case 3:
    printf("3");
    break;
  default:
    printf("Rama falsa");
    break;
}
```

ESTRUCTURA DE CONTROL: FOR

Pseudocodigo

```
Repetir Para (valIni,valFin,paso)
   Imprimir("Valor: ", i)
Fin Repetir Para
```

Standard C

```
for (int i=1; i<11; i++){
  printf("Valor: %d\n", i);
}
```

ESTRUCTURA DE CONTROL: WHILE

Pseudocodigo

```
Repetir Mientras(condicion)
   Imprimir(“Loop infinito")
Fin Repetir Mientras
```

Standard C

```
while (1){
  printf("Loop infinito");
}
```

ARREGLOS

Pseudocodigo

```
Tipo Estructurado
   vec = arreglo(10) entero
   mat = arreglo(10,5) entero
Variables
   vNumeros: vec;
   mNumeros: mat;
```

Standard C

```
int vNumeros[10];
int mNumeros[10][5];
```

REGISTROS

Pseudocodigo

```
Persona = registro
   nombre: carácter[15]
   apellido: carácter[15]
   edad: entero[3]
Fin Registro


Variables
   p1:Persona
```

Standard C

```
struct Persona{
        char nombre[15];
        char apellido[15];
        int edad;
};

struct Persona p1;
```

PROCEDIMIENTOS

```
//-- Módulos prototipados o definidos antes de Main
void moduloA (unParam, otroParam); //Modulo prototipado
void moduloB (unParam, otroParam){ //Modulo definido
  [bloque de código del modulo]
}
//-- Main
int main(){
  [bloque de código de main]
}
//-- Definición de módulos prototipados
void moduloA (unParam, otroParam){

  [bloque de código del modulo]

}
```

PASAJE DE PARAMETROS

```
//Procedimiento que recibe dos números y devuelve la suma en "resul"
void unProcedimiento (int param1, int param2, int *resul){
  *resul = param1 + param2;   //Para usar el puntero, uso asterisco
}

int x, y, z;

int main(){
  //..
  unModulo (x,y,&z); //Invoco al procedimiento
  //..
}
```

- Param1 y param2 son parámetros recibidos por valor.
- Resul es un parámetro que se recibe por referencia.

PASAJE DE PARAMETROS - REGISTRO

```
//Pasaje por referencia
void cargarPersona (struct Persona *reg){
    strcpy(reg->apellido, “Apellido");
    strcpy(reg->nombre, “Nombre");
    reg->edad= 24;
}

//Pasaje por valor
void mostrarPersona (struct Persona reg){
    printf("Datos de la persona\n-------------------\n\n");
    printf("Nombre: %s\n", reg.nombre);
    printf("Apellido: %s\n", reg.apellido);
    printf("edad: %d\n", reg.edad);
    _getch();
}
```

FUNCIONES

```
//Función tipo entero que recibe dos números y retorna la
suma
int unaFuncion (int param1, int param2){
  return param1 + param2;  //Retorno el resultado
}

int x, y, z;

int main(){
  //..
  z = unaFuncion (x,y); //Invoco a la función
  //..
}
```

LISTAS

```
struct Producto{ //Defino Producto, el registro de datos
    int codigo;
    char nombre[15];
    int stock;
};

struct Lista{           //Defino el registro de la lista
    struct Producto dato;
    struct Lista *psig;
};

struct Lista *L;     //Declaro una variable tipo Lista
```

NUEVO NODO DE LISTA

```
struct Lista *nuevo=malloc(sizeof(struct Lista));
```

PASAJE DE LISTA POR PARAMETRO

```
//Pasaje por referencia
void porReferencia(struct TipoReg **lista){
    .
    .
}
//Invocacion
porReferencia(&lista);

//Pasaje por copia
void porCopia(struct TipoReg *lista){
    .
    .
}
//Invocacion
porCopia(lista);
```

PILA

```
struct Producto{ //Defino Producto, el registro de datos
    int codigo;
    char nombre[15];
    int stock;
};

struct Pila{              //Definicion de pila
    struct Producto dato;
    struct Pila *psig;
};

struct Pila *pila;   //Declaro una variable tipo Lista
```

COLA

```
struct Proceso{              //Registro del proceso que almacena la cola
    int id;
    char nombre[20];
    char prioridad[15];
    char estado[15];
};
struct lCola{                //Definición de la lista de la cola
    struct Proceso dato;
    struct lCola *psig;
};
struct Cola{                 //El registro que maneja los punteros de cola
    struct lCola *pini; // Puntero inicial
    struct lCola *pfin; // Puntero final
};
```

RESUMEN

```
//Librerias a usar
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

//Declaracion de variables
int entero;
char caracter;
char string[tamaño];
float real;

//Funciones para operar strings
strcpy([destino],[fuente]);
strcmp([primerString],[segundoString]);
```

```
//Definir un registro
struct nomRegistro{
   [campos]
}

//Declaracion de variables
int entero;
char caracter;
char string[tamaño];
float real;

//Definir un nodo
struct [TipoReg]
*NomVar=malloc(sizeof(struct [TipoReg]));
```

```
//Pasaje por copia y por referencia de un elemeno NO LISTA
void porReferencia([tipoDato] *nomVariable);
porReferencia(&variable);

//Pasaje por copia y por referencia de una lista
void porReferencia(struct TipoReg **lista);
porReferencia(&lista);
```

```
void porCopia([tipoDato] lista);
   porCopia(variable);

void porCopia(struct TipoReg *lista);
   porCopia(lista);
```