1. Introducción a la programación en c

- C fue diseñado originalmente en 1972para el SO UNIX en el DEC PDP-11 por Dennis Ritchie en los Laboratorios Bell.
- El primer libro de referencia de c es: The c Programming Languaje (1978)
- 1989 aparece el estándar ANSI C.
- 1990 aparece el estándar ISO C (actual estándar de C)

2. Fundamentos de C

- // doble diagonal indica comentario en una línea
- Main.- Escribiremos el código principal de nuestro programa
- Variables = utilizado para representar un cierto tipo de información (las de abajo)
  Puede almacenar diferentes valores en distintas partes del programa.
- Char carácter = %s
- Int entero = %d
- Float reales = %f
- Double flotante largo
- Expresiones y sentencias = representa una unidad de datos simple, tal como un numero o carácter.
  Ejem: int **a = 1**, float **n = 4.0**
- Printf = Permite imprimir información por la salida estándar (monitor)
- Scanf = permite leer datos del usuario
- Define, Define constantes simbólicas en el programa
- Operadores de incremento o decremento, ++ incremento en uno el operador, -- decremento en uno el operador .- i++ , i—

3. Herramientas de desarrollo

- Compilación de un programa, código fuente – código ensamblador – código objeto – código ejecutable

4. Instrucciones sentencias de control

- Sección if: Se utiliza para elegir entre cursos alternativos de acción
  Si la expresión 1 es verdadera se ejecuta la sentencia siguiente, si la condición resulta falsa se ignora las sentencias siguientes y se ejecuta el siguiente fragmento de código
- If – else: Elige entre cursos alternativos de acción.
- Switch (menú): Forma decisiones que proporciona una estructura de selección múltiple switch, esta formada por etiquetas **case**.
  Break: romperá la ejecución de los casos, regresando el control fuera de switch, con eso impide que se ejecuten todos los casos.

Instrucción de iteración

- Ciclos: Grupo de instrucciones que la computadora ejecuta en forma repetida.
- While: repetición
- Do – while: se ejecutan las sentencias al menos una vez después se evalúa la expresión

5. Funciones

- En c los módulos se llaman funciones, se pueden escribir funciones para definir tareas especificas
  -Las funciones necesitan ser declaradas.
  -Ser definidas (escribir el código que ejecutara)
  - Ser llamadas (invocarlas dándole los argumentos que necesita)

- Void: Para que la función no regrese ningún valor o si no hay argumentos en compiladores.

Ejemplo. Programa que cuenta los caracteres introducidos desde el teclado hasta que se presiona enter

```
#include<stdio.h>
#include<stdlib.h>

int cuenta_caracteres(void); //prototipo

main()//inicia main
{
 int num_car;
 num_car=cuenta_caracteres();//llamada a la función
 printf("Hay %d caracteres\n",num_car);
 system("pause");
}//fin de main

int  cuenta_caracteres(void)//declaración de función
{
char c;
int cont=0;
c=getchar();

  while(c!='\n')
  {
  cont=cont+1;
  c=getchar();
  }
return cont;
}
```

- No se permite declarar una función dentro de otra

6. Arreglos

- Colección de datos o de variables del mismo tipo, referenciados bajo un mismo nombre y almacenados en localidades consecutivas de memoria.
  -Los arreglos se declaran con variables. Ejem: int A[10]
- Arreglo unidimensional: Almacena n elementos de un mismo tipo y acceder a ellos mediante un mismo índice
  -Los arreglos ocupan memoria y se reservan al momento de declararlos

En este ejemplo se declara una cadena de caracteres denominada “cadena” y reserva espacio para almacenar los siguientes caracteres:

'H' ‘o' 'l' ‘a' '\0'

```
#include<stdio.h>
#include<stdlib.h>

main()
{
 char cadena[]="hola";
 printf("La cadena es %s \n", cadena);
 printf("Los caracteres son:\n");
 printf("%c\n", cadena[0]);
 printf("%c\n", cadena[1]);
 printf("%c\n", cadena[2]);
 printf("%c\n", cadena[3]);
 printf("%c\n", cadena[4]);
system("pause");
}
```

cadena[i], representará el i-esimo carácter de la cadena

- gets: lee caracteres hasta encontrar un retorno en línea (enter) solo para cadenas
- Arreglo multidimensional
  Es la representación de tablas que organizan información en filas y columnas

Programa que llena e imprime un arreglo bidimensional

```
#include<stdio.h>
#include<stdlib.h>

main()
{
 int A[3][3],i,j;                  //Es importante definir el tamaño del arreglo pues es necesario
                  //para reservar espacio en memoria

 for(i=0;i<3;i++)                  //recorrerá los elementos del arreglo desde el elemento 0
   for(j=0;j<3;j++){
   printf("Dame el elemento %d,%d: ",i,j);
   scanf("%d",&A[i][j]);       //en cada iteración se leerá desde el teclado un valor
}

 for(i=0;i<3;i++){                 //Es necesario volver a recorrer el arreglo para leerlo
 printf("\n");
   for(j=0;j<3;j++)
   printf("\tA[%d][%d]= %d ",i,j,A[i][j]);

}
 printf("\n");
  system("pause");           //como solo se afecta a una línea podemos prescindir de {}
 }
```

7. Apuntadores

- El nombre de la variable determina el tipo (char, int, float o double) y la dirección determina donde esta almacenada
- Son variables que contienen direcciones de memoria como sus valores, por lo regular una variable contiene directamente un valor especifico
- **Un apuntador es una variable que contiene la dirección de otra variable y se representa (&).**
  -Apuntador se refiere indirectamente a un valor
  - * permite el acceso a una variable ejem: *nombre
  - se declara antes del int main

- **Memoria dinámica**
  - Por medio de apuntadores se puede reservar o liberar memoria dinámica, para ello se utiliza la función **malloc**
- Malloc :solicita una sección de memoria, Ejem: Char malloc (1000)

Ejemplo Malloc

Programa que llena un arreglo unidimensional usando memoria dinámica

```
#include <stdio.h>
#include <stdlib.h>

main(){

    float *ap;  //Se declara el apuntador ap, el cual hará referencia a datos de tipo flotante
    int n=3,i;

    ap =(float *)malloc(n*sizeof(float));  //Se reserva en memoria el espacio necesario para tres datos de tipo flotante
                        // A partir de este instante el apuntador ap puede ser tratado como un arreglo
    for(i=0;i<n;i++)
    {
    printf("\nCual es peso de tu amigo %d:  ",i);
    scanf("%f", &ap[i]);
    }

    for(i=0;i<n;i++)
     printf("\nLas pesos son\n %f\n",ap[i]);

     system("PAUSE");
     }
```

- Con malloc podremos liberar la memoria en tiempo de ejecución

8. Estructuras

- Conjunto de variables o datos de diferentes tipos
  -Los datos de una estructura se les puede llamar miembros, elementos, campos.

Estructuras

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct amigos    //La estructura se llama amigos
{
char nombre[15];  //elemento de la estructura: nombre
int edad;       //elemento de la estructura: edad
float peso;      //elemento de la estructura: peso
char sexo[5];    //elemento de la estructura: sexo
int cel;        //elemento de la estructura: cel
}a1,a2;         //Variables de tipo estructura: a1,a2
              //Estas dos variables tendran todos los elementos de la estructura
main(){

struct amigos a3; //Se declara una variable de tipo estructura dentro de main()

printf("Se creo la estructura correctamente\n");

printf("\n");
system("PAUSE");

}
```

- Acceso a elementos de una estructura.
  - Los miembros de una estructura se pueden manipular individualmente, Para acceder a elementos de una estructura se utiliza el operador (.) y el operador apuntador de una estructura (->), también llamado operador flecha.

Estructuras

```
//Inicialización de una variable de tipo estructura amigos
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct amigos    //La estructura se llama amigos
{
char nombre[15];  //elemento de la estructura: nombre
int edad;        //elemento de la estructura: edad
float peso;      //elemento de la estructura: peso
char sexo[5];    //elemento de la estructura: sexo
int cel;         //elemento de la estructura: cel
}a1,a2;          //Variables de tipo estructura: a1,a2
              //Estas dos variables tendran todos los elementos de la estructura
main(){

struct amigos a3={"pepe",15,82.31,"m",123456}; //Se declara una variable de tipo estructura dentro de main()

fflush(stdin);   //Limpiar el buffer de entrada del teclado
printf("\n Dame el nombre completo del amigo ");
gets(a1.nombre);   //Adquiere una cadena del teclado y la almacena en el elemento nombre de la variable a1 de tipo amigos
```

- Arreglos de tipo estructura, especificando su tamaño
  -Ejem: struct nombre_struct variable_tipo_struct[100];

9. Archivos

- Concepto lógico que permite almacenar información de modo permanente y acceder y/o alterar la misma cuando sea necesario.
- Los archivos se pueden abrir en los siguientes modos:
  - r+ Abre un archivo para actualizar (leer y escribir)
  - w+ Abre un archivo para actualizar, se crea si no existe, si existe se sobre escribe.
  - a+ Abre un archivo para actualizar al final del contenido, si no existe se crea

Archivos, función de lectura fgetc(apuntador_archivo)

```
#include<stdio.h>
#include<stdlib.h>

main(){
char c;
FILE *ap;
ap=fopen("texto.txt","r");
if(ap==NULL)
printf("No se puede abrir el
archivo\n");
else{
printf("EL archivo se abrio
correctamente\n");
while(feof(ap)==0){
c=fgetc(ap);
printf("\n%c",c);
}
fclose(ap);
system("PAUSE");
}
```

Lee carácter por carácter

Lenguaje de programación iniciando con mi nombre/apellido

Erlang

Erlang es un lenguaje de programación concurrente y un sistema de ejecución que incluye una máquina virtual (BEAM) y bibliotecas.

El subconjunto de programación secuencial de Erlang es un lenguaje funcional, con evaluación estricta, asignación única, y tipado dinámico. Fue diseñado en la compañía Ericsson para realizar aplicaciones distribuidas, tolerantes de fallos, soft-real-time y de funcionamiento ininterrumpido.