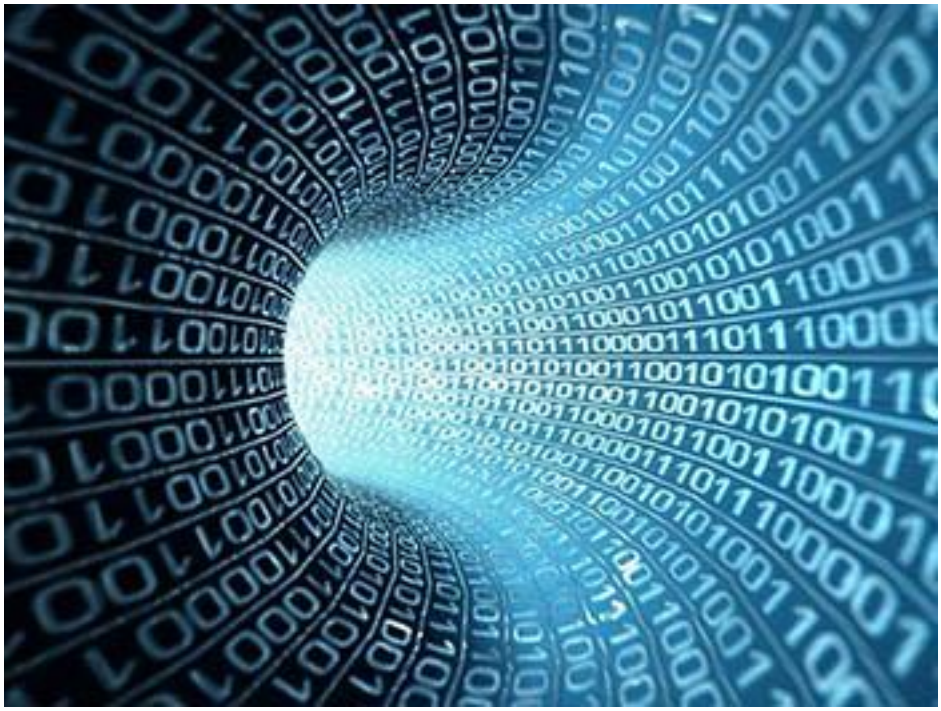
UNIDAD CURRICULAR: ALGORITMICA Y PROGRAMACIÓN

UNIDAD IX. ESTRUCTURAS DE REGISTROS

CONTENIDO:



UNIDAD IX

ESTRUCTURAS DE REGISTROS

DEFINICIÓN: Es una colección de datos relacionada con una entidad particular definida por el usuario como un tipo de datos que se debe declarar antes de ser utilizada. Estos datos se reciben el nombre de campos o elementos de datos .

DECLARACIÓN

```
struct nombre_de_la_estructura
{ tipo nombre_del_campo1;
  tipo nombre_del_campo2;
   ..........
  tipo nombre_del_campon;
}
```

Ejemplo: Declaremos una estructura de registro relacionada con los datos de un articulo:

```
struct articulo
{
 char codigo[12];
 char descripcion[40];
 float precio;
int cant_exist;
char fecha_vencimiento[12];
}
```

Una vez definida la estructura de registro, se declaran las variables de tipo_estructura, tal como se declaran las variables en C++.

Siguiendo el ejemplo, la declaración de las variables de este tipo de estructura se realizaría de la siguiente manera:

```
articulo art1, art2;
```

ACCESO A ESTRUCTURAS

Cuando se accede a una estructura de registro, o bien se almacena información en la estructura o se recupera información de la estructura. Se puede acceder a los campos o elementos de una estructura utilizando el operador punto (.) o el operador puntero (->).Para ello se hace referencia al nombre de la variable tipo registro seguido del operador punto o puntero y el nombre del campo.

Almacenamiento de información en estructuras:

Se puede almacenar información en una estructura de registro, bien sea por asignación directa o por lectura.

Sintaxis:

Utilizando el operador punto (.)

Variable_estructura.nombre_campo = dato;

Utilizando el operador puntero (->):

Variable_estructura->nombre_campo= dato;

Ejemplo:

art1.precio=500;

strcpy (art1.codigo,"00085");

art->precio=500;

Lectura de información de una estructura

Se emplea una instrucción de lectura de datos utilizando el operador punto o puntero.

Siguiendo el ejemplo:

cout<<"Ingrese el precio del artículo:";

cin>>art1.precio;

Utilizando puntero:

cout<<"Ingrese el precio del artículo:";

cin>>art1->precio;

Recuperación de información de una estructura

Se recupera la información de una estructura de registro a través del operador de asignación o empleando una instrucción de salida o escritura de datos utilizando el operador punto o puntero.

Sintaxis:

Utilizando el operador punto (.)

```
variable=Variable_estructura.nombre_campo;
cout<<Variable_estructura.nombre_campo;
```

Ejemplo:

```
Precio_base=art1.precio;
cout<<"El precio es: "<<art1.precio;
```

Utilizando el operador puntero (->):

```
variable=Variable_estructura->nombre_campo;
cout<<Variable_estructura->nombre_campo;
```

Ejemplo:

```
Precio_base=art1->precio;
cout<<"El precio es: "<<art1->precio;
```

ESTRUCTURA Y ARREGLO

Es posible crear un arreglo de estructuras de registros, de esta manera se puede guardar en un arreglo información de distinto tipo.

Declaración

```
Tipo_registro nombre_del_arreglo[número_posiciones];
```

Ejemplo:

```
articulo lista_articulos[50];
```

El acceso a dicho arreglo sería similar a lo visto anteriormente con la salvedad de agregar el índice o posición del arreglo al que se está accediendo.

Ejemplo:

strcpy(lista_articulos [0].codigo,"00008");

lista_articulos [0].precio=500;

EJERCICIOS RESUELTOS

a)Declare una estructura de registro diseñada para manejar la información de materiales bibliográficos de una biblioteca: cota, título, autor, editorial, tipo de material, cantidad total de ejemplares, cantidad de ejemplares disponibles, cantidad de ejemplares prestados.

```
struct material
{
char cota[10];
char titulo[50];
char autor[50];
char editorial[40];
char tipo_material;
int cant_total, cant_dispo, cant_prest;
}
```

b)Tomando en cuenta la estructura anterior, declare la variable *enciclopedia* de tipo *estructura material*.

```
material enciclopedia;
```

c) Ahora declare arreglos que permitan guardar la información de 100 libros y 25 revistas.

```
material libros[100], revistas[25];
```

d) Realice una función que lea los datos que identifican los 100 libros y los almacene en el arreglo. Con respecto a tipo_material, asignar una L. Cant_dispo debe ser igual a la cantidad total y cantidad prestada debe estar inicializada en 0.

```
void cargar_arreglo()
```

```
{int k;

 for (k=0; k<100; k++)

{ cout<<"Ingrese la cota del libro:";
  cin>>libros[k].cota;
  cout<<"Ingrese el título:";
  cin>>libros[k].titulo;
  cout<<"Ingrese el autor del libro:";
  cin>>libros[k].autor;
  cout<<"Ingrese la editorial:";
  cin>>libros[k].editorial;
  cout<<"Ingrese la cantidad total de ejemplares:";
  cin>>libros[k].cant_total;
  libros[k].tipo_material='L';
  libros[k].cant_dispo= libros[k].cant_total;
  libros[k].cant_prest=0;
}
}
```

e) Realice una función que muestre la cota y el título de los libros disponibles para préstamo en biblioteca.

```
void mostrar_disponibles()

{int k;

 for (k=0; k<100; k++)

  { if (  libros[k].cant_dispo>0)
     { cout<< "Cota: " <<libros[k].cota;
       cout<< "Título: " <<libros[k].titulo;
     }
   }
}
```

REFERENCIAS BIBLIOGRÁFICAS

Bassard, G y Bratley, P. (2010). Fundamentos de algoritmia. Prentice-Hall.

Joyanes, L. (2008). Fundamentos de programación. Algoritmos , Estructuras de datos y objetos. Mc Graw Hill. Tercera edición.

Joyanes, L. y Zahonero, I. (2005). Programación en C. Metodología, algoritmos y Estructura de datos. Mc Graw Hill. Segunda Edición

Martí, N. y Ortega, Y. (2004). Estructuras de datos y Métodos Algorítmicos. Ejercicios Resueltos. Pearson Education.