



Estructuras

MG. MARCELINO TORRES VILLANUEVA

ESTRUCTURAS (struct)

Una estructura (struct) es un tipo de dato definido por el usuario. Las estructuras son similares a los registros utilizados en otros lenguajes de programación.

Una estructura es un tipo de dato estructurado que consta de un conjunto de elementos que pueden ser del mismo tipo o de tipos diferentes.

Los componentes de una estructura se denominan campos. Cada campo tiene un nombre llamado identificador de campo que es algún identificador elegido por el programador cuando se declara la estructura.

Definición de una estructura

La sintaxis para definir una estructura es la siguiente:

```
struct nombreEstructura{  
    TipodeDato1 campo1;  
    TipoDeDato2 campo2;  
    TipoDeDato3 campo3;  
    ....  
  
    TipoDeDaton campon;  
};
```

Lo que se esta definiendo es un Nuevo tipo de Dato llamado nombreEstructura que tiene un conjunto de campos.

ejemplos

- Declarar una estructura llamada Producto con los siguientes campos : codigo, nombre, precio, stock.

```
struct Producto{  
    char codigo[10];  
    char nombre[60];  
    float precio;  
    int stock;  
};
```

- Declarar la Estructura Punto con campos x e y.

```
struct Punto{  
    float x;  
    float y;  
};
```

Variables tipo estructura

Al declarar la estructura se ha creado un nuevo tipo de dato, pero no se ha declarado variables de ese tipo.

Para declarar variables se puede hacer de la siguiente manera:

- Declarar 3 variables de tipo Producto

```
Producto prod1,prod2, prod3;
```

- Declarar 2 variables de tipo Punto

```
Punto punto1,punto2;
```

Acceso a los campos de una estructura

El acceso a los miembros de una estructura se realiza mediante el operador punto (.)

Por ejemplo si declaramos la variable X de tipo producto

Producto X;

Para acceder a sus campos se hace de la siguiente manera:

```
strcpy(X.codigo, "pr001");  
strcpy(X.nombre, "ace");  
X.precio = 3.5;  
X.stock = 200;
```

Operaciones sobre estructuras

- Una operación que se puede realizar entre estructuras es la asignación (copia del contenido de una estructura en otra estructura del mismo tipo). Si A y D son variables tipo estructura del mismo tipo, la sentencia

A=D;

Copia todo los valores asociados de la variable D a la variable A.

Ejemplo:

Punto P1, P2;

P1.x=10;

P1.y=20;

La sentencia de asignación

P2=P1;

Equivale a :

P2.x = P1.x;

P2.y=P1.y;

Operaciones sobre estructuras

- No se puede comparar completamente una variable tipo estructura con otra. Por ejemplo si se tiene dos variables A y B de tipo estructura

Punto A, B;

No se puede hacer esto:

```
if(A==B)
```

Si se quiere comparar se tiene que hacer campo por campo

```
If(A.x==B.x && A.y==B.y)
```


Estructuras anidadas

Es posible declarar una estructura con campos que sean otras estructuras. Una estructura con uno o mas campos que son tipo estructura se llaman registro jerárquico o anidado.

Ejemplo:

```
struct NombreEmp{
    char apellidos[30];
    char nombres[30];
};
struct CalleNumero{
    char calle[40];
    int numero;
};
struct Empleado{
    NombreEmp nombre;
    CalleNumero dir;
};
```

Acceso a los registros anidados

Para referenciar un campo en estructuras anidadas se debe indicar el camino a seguir en orden jerárquico desde el nombre de la estructura raíz hasta el campo específico. Ejemplo:

Empleado E;

```
strcpy(E.nombre.apellidos, "Estela");
```

```
strcpy(E.nombre.nombres, "Walter");
```

```
strcpy(E.dir.calle, "Zela");
```

```
E.dir.numero= 254
```

Arreglo de estructuras

Si deseamos declarar un arreglo de Estructuras de tipo Producto.

```
struct Producto{  
    char codigo[10];  
    char nombre[60];  
    float precio;  
    int stock;  
};  
Producto V[50];
```

Declaramos un arreglo de 50 elementos de tipo Producto.

Por ejemplo si queremos colocar datos a algún elemento del arreglo lo haríamos:

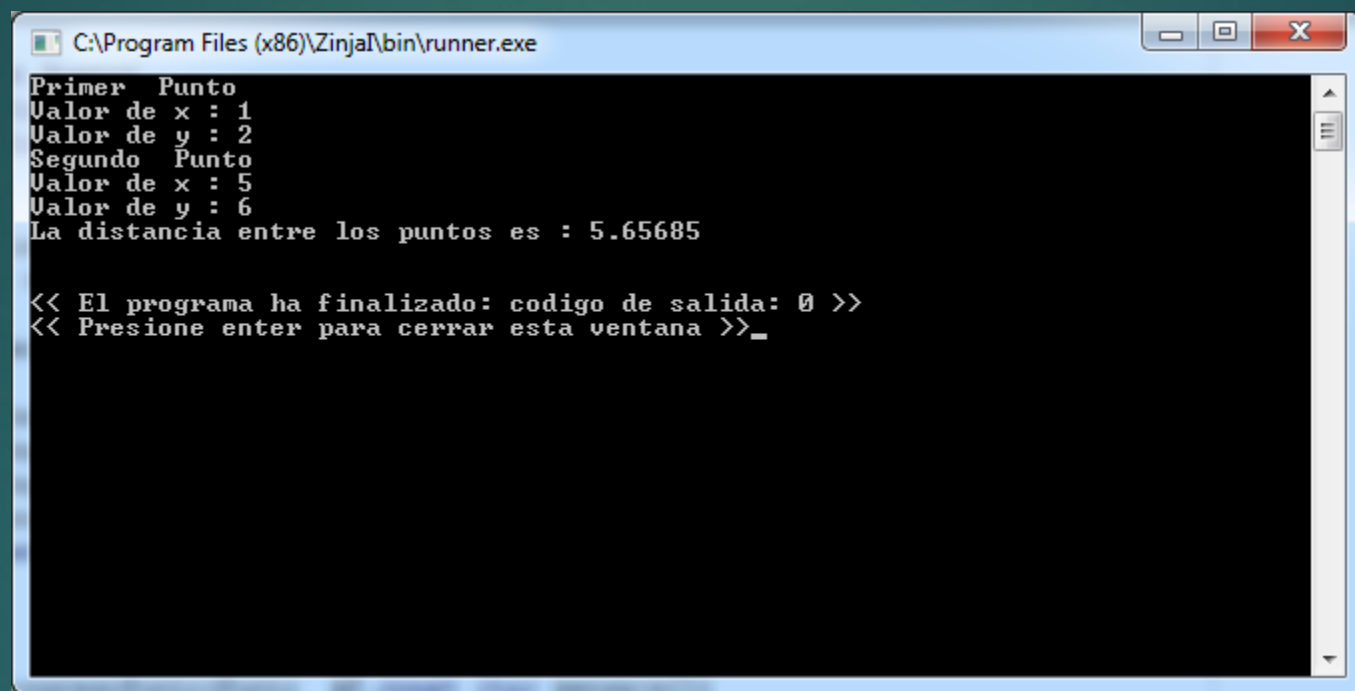
`V[i].codigo, V[i].nombre, V[i].precio, V[i].stock`

EJERCICIOS RESUELTOS

1. Programa para ingresar las coordenadas de 2 puntos del plano cartesiano. Reportar la distancia entre ellos.

```
1  #include<iostream>
2  #include<math.h>
3  using namespace std;
4
5  struct Punto{
6      float x;
7      float y;
8  };
9
10 void ingresoPunto(Punto &P,const char mensaje[]);
11 float distancia(Punto P1, Punto P2);
12
13 int main()
14 {
15     Punto P1, P2;
16     ingresoPunto(P1,"Primer Punto");
17     ingresoPunto(P2,"Segundo Punto");
18     cout<<"La distancia entre los puntos es : "<<distancia(P1,P2)<<endl;
19     return 0;
20 }
```

```
21  
22  
23 void ingresoPunto(Punto &P,const char mensaje[])  
24 {  
25     cout<<mensaje<<endl;  
26     cout<<"Valor de x : ";  
27     cin>>P.x;  
28     cout<<"Valor de y : ";  
29     cin>>P.y;  
30 }  
31  
32  
33 float distancia(Punto P1, Punto P2)  
34 {  
35     return sqrt(pow (P1.x-P2.x,2)+pow (P1.y-P2.y,2));  
36 }
```



```
C:\Program Files (x86)\Zinja\bin\runner.exe

Primer Punto
Valor de x : 1
Valor de y : 2
Segundo Punto
Valor de x : 5
Valor de y : 6
La distancia entre los puntos es : 5.65685

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>_
```

2. Hacer un programa para ingresar los nombres y las notas de los alumnos de Fundamentos de Programación y se reporte:

- a) Una Lista en orden Alfabético
- b) Una Lista en orden de Merito.

```
1  #include<iostream>
2  #include<string.h>
3
4  using namespace std;
5
6  struct Alumno{
7      char nombre[40];
8      float nota;
9  };
10
11 void numAlumnos(int &n);
12 void ingreso(Alumno A[], int n);
13 void reporte(Alumno A[], int n);
14 void ordenAlfabetico(Alumno A[], int n);
15 void ordenMerito(Alumno A[], int n);
16
```

```
17  int main()
18  {
19      Alumno A[100];
20      int n;
21      numAlumnos(n);
22      ingreso(A,n);
23      ordenAlfabetico(A,n);
24      cout<<"Lista en orden alfabetico"<<endl;
25      reporte(A,n);
26      ordenMerito(A,n);
27      cout<<"Lista en orden de Merito"<<endl;
28      reporte(A,n);
29      return 0;
30  }
31
32
33  void numAlumnos(int &n)
34  {
35      do{
36          cout<<"Numero de alumnos : ";
37          cin>>n;
38      }while(n<=0);
39  }
40
```



```
41 void ingreso(Alumno A[], int n)
42 {
43     int i;
44     for(i=0;i<n;i++)
45     {
46         cout<<"Datos del alumno "<<i+1<<endl;
47         cout<<"Nombre : ";
48         cin.ignore();
49         cin.getline(A[i].nombre,40);
50         do{
51             cout<<"Nota : ";
52             cin>>A[i].nota;
53         }while(A[i].nota<0 || A[i].nota>20);
54     }
55 }
56
57
58 void reporte(Alumno A[], int n)
59 {
60     int i;
61     for(i=0;i<n;i++)
62         cout<<A[i].nombre<<"\t " <<A[i].nota<<endl;
63 }
```



```
C:\Program Files (x86)\Zinja\bin\runner.exe

Numero de alumnos : 4
Datos del alumno 1
Nombre : Luis
Nota : 15
Datos del alumno 2
Nombre : Ana
Nota : 16
Datos del alumno 3
Nombre : Jorge
Nota : 18
Datos del alumno 4
Nombre : Cecila
Nota : 15
Lista en orden alfabetico
Ana      16
Cecila   15
Jorge    18
Luis     15
Lista en orden de Merito
Jorge    18
Ana      16
Cecila   15
Luis     15

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

Ejercicios resueltos

- 1) Ingresar n valores y sus Respectivos pesos y calcular el Promedio Ponderado.

$$\bar{x} = \frac{\sum_{i=1}^n x_i w_i}{\sum_{i=1}^n w_i} = \frac{x_1 w_1 + x_2 w_2 + x_3 w_3 + \dots + x_n w_n}{w_1 + w_2 + w_3 + \dots + w_n}$$

donde:

x_i : Valores

w_i : Pesos

```
1  #include <iostream>
2  using namespace std;
3
4  struct Dato{
5      float valor;
6      float peso;
7  };
8
9  void numDatos(int &n);
10 void ingresoDatos(Dato v[], int n);
11 void reporteDatos(Dato v[], int n);
12 float promedioPonderado(Dato v[], int n);
13
14 int main(int argc, char *argv[]) {
15     Dato v[100];
16     int n;
17     numDatos(n);
18     ingresoDatos(v,n);
19     reporteDatos(v,n);
20     cout<<" El promedio Ponderado es: "<<promedioPonderado(v,n)<<endl;
21     return 0;
22 }
23
```

```

24 void numDatos(int &n)
25 {
26     do{
27         cout<<"Numero de datos : ";
28         cin>>n;
29     }while (n<=0) ;
30 }
31
32 void ingresoDatos(Dato v[], int n)
33 {
34     int i;
35     for(i=0;i<n;i++)
36     {
37         cout<<"Valor ["<<i<<"]:";
38         cin>>v[i].valor;
39         cout<<"Peso["<<i<<"]:";
40         cin>>v[i].peso;
41     }
42 }
43
44 void reporteDatos(Dato v[], int n)
45 {
46     int i;
47     cout<<"Datos Ingresados"<<endl;
48     for(i=0;i<n;i++)
49         cout<<v[i].valor<<" , "<<v[i].peso<<endl;
50 }
51

```

```
52 float promedioPonderado(Dato v[], int n)
53 {
54     int i;
55     float s1=0,s2=0;
56     for(i=0;i<n;i++)
57     {
58         s1 = s1 + v[i].valor*v[i].peso;
59         s2 = s2 + v[i].peso;
60     }
61     return s1/s2;
62 }
```



2) Ingresar la información de n Empleados (código, Apellidos, nombres, teléfono, salario). Luego hacer lo siguiente:

- Dado un código mostrar los datos de un Empleado.
- Dado un código, eliminar el empleado
- Ordenar por Salario descendientemente
- Mostrar Todos los empleados ingresados
- Mostrar el promedio de los salarios


```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4
5  struct Empleado{
6      char codigo[10];
7      char apellidos[40];
8      char nombres[40];
9      char telefono[20];
10     float salario;
11 };
12
13 void numEmpleados(int &n);
14 void ingresoEmpleados(Empleado v[], int n);
15 void reporteEmpleados(Empleado v[], int n);
16 int busqueda(Empleado v[], int n, char codigo[]);
17 void consultarEmpleado(Empleado v[], int n);
18 void eliminarEmpleado(Empleado v[], int &n);
19 void ordenarPorSalario(Empleado v[], int n);
20 float promedioSalarios(Empleado v[], int n);
21
```

```

22 int main(int argc, char *argv[]) {
23     int op,n;
24     Empleado v[100];
25     numEmpleados(n);
26     ingresoEmpleados(v,n);
27     do{
28         system("cls");
29         cout<<"Menu de Empleados"<<endl;
30         cout<<"[1] Consultar Empleado"<<endl;
31         cout<<"[2] Eliminar Empleado "<<endl;
32         cout<<"[3] Ordenar por Salario descendente"<<endl;
33         cout<<"[4] Mostrar Empleados"<<endl;
34         cout<<"[5] Promedio de Sueldos "<<endl;
35         cout<<"[6] Salir "<<endl;
36         cout<<"Ingrese opcion(1-6): ";
37         cin>>op;
38         switch(op)
39         {
40             case 1: consultarEmpleado(v,n);break;
41             case 2: eliminarEmpleado(v,n);break;
42             case 3: ordenarPorSalario(v,n);
43                 system("cls");
44                 cout<<"Datos ordenados por salario"<<endl;
45                 reporteEmpleados(v,n);
46                 break;
47             case 4: system("cls");
48                 reporteEmpleados(v,n);
49                 system("pause");
50                 break;
51             case 5: system("cls");
52                 cout<<"El promedio de sueldos es : "<<promedioSalarios(v,n)<<endl;
53                 system("pause");
54         }
55     }while(op!=6);
56     return 0;
57 }

```

```
58
59 void numEmpleados(int &n)
60 {
61     do{
62         cout<<"Numero de Empleados : ";
63         cin>>n;
64     }while(n<=0);
65 }
66
67 void ingresoEmpleados(Empleado v[], int n)
68 {
69     int i;
70     for(i=0;i<n;i++)
71     {
72         cout<<"Datos del Empleado "<<i+1<<endl;
73         cout<<"codigo :";
74         cin>>v[i].codigo;
75         cin.ignore();
76         cout<<"apellidos : ";
77         cin.getline(v[i].apellidos,40);
78         cout<<"nombres : ";
79         cin.getline(v[i].nombres,40);
80         cout<<"telefono : ";
81         cin>>v[i].telefono;
82         cout<<"sueldo : ";
83         cin>>v[i].salario;
84     }
85 }
86
```

```
87 void reporteEmpleados(Empleado v[], int n)
88 {
89     int i;
90
91     cout<<"Lista de Empleados "<<endl;
92     for(i=0;i<n;i++)
93     {
94         cout<<v[i].codigo<<"", "<<v[i].apellidos<<"", "<<v[i].nombres<<"", "<<v[i].telefono<<"", "<<v[i].salario<<endl;
95     }
96 }
97
98 int buscarCodigo(Empleado v[], int n, char codigo[])
99 {
100     int i;
101     for(i=0;i<n;i++)
102     {
103         if(strcmp(v[i].codigo, codigo)==0)
104             return i;
105     }
106     return -1;
107 }
108
```

```
09 void consultarEmpleado(Empleado v[], int n)
10 {
11     int p;
12     char codigo[10];
13     system("cls");
14     cout<<"Ingrese codigo : ";
15     cin>>codigo;
16     p=buscarCodigo(v,n,codigo);
17     if(p!=-1)
18     {
19         cout<<"Datos del empleado "<<endl;
20         cout<<"Apellidos : "<<v[p].apellidos<<endl;
21         cout<<"Nombres : "<<v[p].nombres<<endl;
22         cout<<"Telefono : "<<v[p].telefono<<endl;
23         cout<<"Sueldo : "<<v[p].salario;
24     }
25     else
26         cout<<"Codigo no se encuentra"<<endl;
27     system("pause");
28 }
29
```

```
30 void eliminarEmpleado(Empleado v[], int &n)
31 {
32     int p;
33     char codigo[10];
34     system("cls");
35     cout<<"Ingrese codigo : ";
36     cin>>codigo;
37     p=buscarCodigo(v,n,codigo);
38     if(p!=-1)
39     {
40         for(int i=p;i<n-1;i++)
41             v[i]=v[i+1];
42         n=n-1;
43         cout<<"dato eliminado"<<endl;
44     }
45     else
46         cout<<"Codigo no se encuentra"<<endl;
47     system("pause");
48 }
49
```

```
150 void ordenarPorSalario(Empleado v[], int n)
151 {
152     int i,j;
153     Empleado temp;
154     for(i=0;i<n-1;i++)
155         for(j=i+1;j<n;j++)
156             if(v[i].salario<v[j].salario)
157             {
158                 temp=v[i];
159                 v[i]=v[j];
160                 v[j]=temp;
161             }
162     }
163 }
164
165 float promedioSalarios(Empleado v[], int n)
166 {
167     int i;
168     float s=0;
169     for(i=0;i<n;i++)
170         s=s+v[i].salario;
171     return s/n;
172 }
```