Estructuras

Una estructura es un grupo de variables las cuales pueden ser de diferentes tipos sostenidas o mantenidas juntas en una sola unidad. La unidad es la estructura.

Las estructuras de datos se emplean con el objetivo principal de organizar los datos contenidos dentro de la memoria de la PC. Así, nuestra primera experiencia con estructuras comienza desde el momento mismo en que usamos en nuestros programas variables de tipos primitivos (`char, short, int, float, etc`).

Sintaxis

En C/C++ se forma una estructura utilizando la palabra reservada `struct,` seguida por un campo etiqueta opcional, y luego una lista de miembros dentro de la estructura. La etiqueta opcional se utiliza para crear otras variables del tipo particular de la estructura:

```
struct [<nombre tipo de estructura>] {
      [<tipo> <nombre_variable[,nombre_variable,...]>;
      [<tipo> <nombre_variable[,nombre_variable,...]>;
      ...
}[<variables de estructura>];
```

Un punto y coma finaliza la definición de una estructura puesto que ésta es realmente una sentencia C/C++.

Sintaxis: Caso uno, estructura anónima

De acuerdo con la sintaxis general de la orden struct es posible crear estructuras de datos anónimas. Solamente hay que tener en cuenta que en una declaración anónima se debe definir al menos una variable al final de la declaración. Por ejemplo, con el siguiente fragmento de código:

```
struct {
      int a;
      int b;
}p1;
```

Se declara y define la variable estructurada `p1`, misma que se compone por los miembros `a` y `b`; ambos del tipo `int`.

Ahora bien, la sintaxis mostrada no es tan común ni conveniente, ya que con la misma solamente se está creando una variable estructurada pero no un nuevo tipo. Es decir, si deseáramos tener otra variable que tuviera las mismas características que posee la variable p1, necesitaríamos escribir exactamente la misma instrucción, salvo que cambiando el nombre de la variable. Por ejemplo:

```
struct {
      int a;
      int b;
}p2;
```

Por supuesto, en una misma línea de instrucción podemos definir más de una variable.

Ejemplo:

```
struct {
      int a;
      int b;
}p1, p2;
```

Entonces, para crear nuevos tipos con struct deberemos de modificar la sintaxis mostrada en los ejemplos anteriores.

Sintaxis: Caso dos, estructura con nombre

Observe que, la sintaxis para declarar estructuras con nombre es bastante parecida a la sintaxis para declarar estructuras anónimas; salvo que una declaración de estructura con nombre se debe especificar el nombre deseado para la misma. Además, en una declaración de estructura con nombre la o las variables definidas al final de la misma son opcionales.

```
struct pareja {
      int a;
      int b;
}p1;
```

En el fragmento de código anterior se declara la estructura identificada como pareja, misma que se compone de los miembros a y b, ambos de tipo int. En el mismo ejemplo, se define la variable p1; la cual es una variable estructurada de tipo pareja.

Ejemplos:

Declarando la variable de estructura `producto1`

```
#include <iostream>
#include <conio.h>

using namespace std;

struct Productos{
    int cod_prod;
    char nom_prod[20];
    float vlr_prod;
    char uni_med_prod[10];
}producto1 = {1020,"Naranja",3500.5,"Libra"};

int main()
{
    cout<<"\n --- IMPRIMIR PRODUCTO --- ";
    cout<<"\n\n El código del producto es : "<<producto1.cod_prod<<endl;
    cout<<" El nombre del producto es : "<<producto1.nom_prod<<endl;
    cout<<" El valor del producto es : "<<producto1.vlr_prod<<endl;
    cout<<" La unidad de medida del producto es : "<<producto1.uni_med_prod<<endl;

    getch();

    return 0;
}
```

Si se desea ingresar más de un producto tendríamos que declarar tantas variables como productos a ingresar declaramos las variables de estructura `producto1, producto2`.

```
#include <iostream>
#include <conio.h>

using namespace std;

struct Productos{
    int cod_prod;
    char nom_prod[20];
    float vlr_prod;
    char uni_med_prod[10];
}producto1 = {1020,"Naranja",3500.5,"Libra"}, producto2 = {2040,"Lulo",2800.5,"Libra"};

int main()
{
    cout<<"\n --- IMPRIMIR PRODUCTO --- ";
    cout<<"\n\n El código del producto es : "<<producto1.cod_prod<<endl;
    cout<<" El nombre del producto es : "<<producto1.nom_prod<<endl;
    cout<<" El valor del producto es : "<<producto1.vlr_prod<<endl;
    cout<<" La unidad de medida del producto es : "<<producto1.uni_med_prod<<endl;

    getch();

    return 0;
}
```

Declarando la variable de estructura `producto1` en la función principal `main()`.

```
#include <iostream>
#include <conio.h>

using namespace std;

struct Productos{
    int cod_prod;
    char nom_prod[20];
    float vlr_prod;
    char uni_med_prod[10];
};

int main()
{
    struct Productos producto1 = {1020,"Naranja",3500.5,"Libra"};

    cout<<" --- IMPRIMIR PRODUCTO --- ";
    cout<<"\n\n El código del producto es : "<<producto1.cod_prod<<endl;
    cout<<" El nombre del producto es : "<<producto1.nom_prod<<endl;
    cout<<" El valor del producto es : "<<producto1.vlr_prod<<endl;
    cout<<" La unidad de medida del producto es : "<<producto1.uni_med_prod<<endl;

    getch();

    return 0;
}
```

Para no tener que crear n números de variables, se podría acudir a aplicar estructuras con arreglos.

Estructuras con arreglos

Así como bien podemos crear un array de cualquier tipo de dato estándar C++, también podemos crear un array de estructuras, después de todo, recordemos que una estructura lo que hace es crear un nuevo tipo de dato.

Los arrays de estructuras son idóneos para registrar un archivo completo de empleados, un archivo de inventario o cualquier otro dato que tome el formato de la estructura.

```
#include <iostream>
#include <conio.h>

using namespace std;

struct Productos{
    int cod_prod;
    char nom_prod[20];
    float vlr_prod;
    char uni_med_prod[10];
}producto1[2] = {1020, "Naranja", 3500.5, "Libra", 2040, "Lulo", 2800.8, "Libra"};


int main()
{

    cout<<" --- IMPRIMIR PRODUCTO # 1 ---";
    cout<<"\n\n El código del producto es : "<<producto1[1].cod_prod<<endl;
    cout<<" El nombre del producto es : "<<producto1[1].nom_prod<<endl;
    cout<<" El valor del producto es : "<<producto1[1].vlr_prod<<endl;
    cout<<" La unidad de medida del producto es : "<<producto1[1].uni_med_prod<<endl;

    cout<<" --- IMPRIMIR PRODUCTO # 2 ---";
    cout<<"\n\n El código del producto es : "<<producto1[2].cod_prod<<endl;
    cout<<" El nombre del producto es : "<<producto1[2].nom_prod<<endl;
    cout<<" El valor del producto es : "<<producto1[2].vlr_prod<<endl;
    cout<<" La unidad de medida del producto es : "<<producto1[2].uni_med_prod<<endl;


    getch();

    return 0;
}
```

Para hacerlo más funcional, vamos a optimizar el código, se pedirá que el usuario ingrese los datos, y utilizaremos estructuras repetitivas para el ingreso e impresión de los datos.

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>

using namespace std;

struct Productos{
    int cod_prod;
    char nom_prod[20];
    float vlr_prod;
    char uni_med_prod[10];
}producto1[2];


int main()
{
    for (int i=0;i<2;i++){

        cout<<"\n --- INGRESO PRODUCTO --- "<<"# "<<i+1;
        cout<<"\n\n Digite el código del producto : ";
        cin>>producto1[i].cod_prod;
        cout<<" Digite el nombre del producto : ";
        cin>>producto1[i].nom_prod;
        cout<<" Digite el valor del producto : ";
        cin>>producto1[i].vlr_prod;
        cout<<" Digite la unidad de medida del producto : ";
        cin>>producto1[i].uni_med_prod;

        system("cls");
    }

    for (int i=0;i<2;i++){

        cout<<"\n --- IMPRIMIR PRODUCTO --- "<<"# "<<i+1;
        cout<<"\n\n El código del producto es : "<<producto1[i].cod_prod<<endl;
        cout<<" El nombre del producto es : "<<producto1[i].nom_prod<<endl;
        cout<<" El valor del producto es : "<<producto1[i].vlr_prod<<endl;
        cout<<" La unidad de medida del producto es : "<<producto1[i].uni_med_prod<<endl;
    }

    getch();

    return 0;
}
```

De igual forma podemos anidar estructuras, las cuales nos permiten por ejemplo tener datos de un mismo empleado pero en estructuras independientes y relacionadas entre si.

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>

using namespace std;

struct info_laboral{
    char dep_emp[20];
    int sal_emp;
    char car_emp[20];
};

struct info_empleado{
    int doc_emp;
    char nom_emp[20];
    int eda_emp;
    struct info_laboral lab_emp;
}empleado[2];


int main()
{
    for (int i=0;i<2;i++){

        cout<<"\n --- INGRESO DATOS DEL EMPLEADO --- "<<"# "<<i+1;
        cout<<"\n\n Digite el documento del empleado : ";
        cin>>empleado[i].doc_emp;
        cout<<" Digite el nombre del empleado : ";
        cin>>empleado[i].nom_emp;
        cout<<" Digite la edad del empleado : ";
        cin>>empleado[i].eda_emp;
        cout<<" Digite el departamanto del empleado : ";
        cin>>empleado[i].lab_emp.dep_emp;
        cout<<" Digite el salario del empleado : ";
        cin>>empleado[i].lab_emp.sal_emp;
        cout<<" Digite el cargo del empleado : ";
        cin>>empleado[i].lab_emp.car_emp;

        system("cls");
    }

    for (int i=0;i<2;i++){

        cout<<"\n --- IMPRIMIR DATOS DEL EMPLEADO --- "<<"# "<<i+1;
        cout<<"\n\n El documento del empleado es : "<<empleado[i].doc_emp<<endl;
        cout<<" El nombre del empleado es : "<<empleado[i].nom_emp<<endl;
        cout<<" La edad del empleado es : "<<empleado[i].eda_emp<<endl;
        cout<<" El departamento del empleado es : "<<empleado[i].lab_emp.dep_emp<<endl;
        cout<<" El salario del empleado es : "<<empleado[i].lab_emp.sal_emp<<endl;
        cout<<" El cargo del empleado es : "<<empleado[i].lab_emp.car_emp<<endl;
    }

    getch();

    return 0;
}
```