Estructuras Compuestas

Variables Primitivas

Sabemos en un principio que contamos con variables primitivas.

Test:

https://replit.com/join/mbvkvmhpzj-maximo-arielari

Nombre	Descripción	Tamaño*	Rango de valores*
`char`	Carácter o entero pequeño.	1byte	con signo: -128 to 127 sin signo: 0 a 255
`short int (short)`	Entero corto.	2bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
`int`	Entero.	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
`long int (long)`	Entero largo.	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
`bool`	Valor booleano. Puede tomar dos valores: verdadero ofalso.	1byte	true o false
`float`	Número de punto flotante.	4bytes	3.4e +/- 38 (7 digitos)
`double`	De punto flotante de doble precisión.	8bytes	1.7e +/- 308 (15 digitos)
`long double`	Long de punto flotante de doble precisión.	8bytes	1.7e +/- 308 (15 digitos)

Tiempo de vida de un valor Local

Las variables definidas Localmente tienen un tiempo de vida reducido a su ciclo de funcionamiento. Cada valor es descartado cuando la función culmina .

Test https://replit.com/join/bonyukppje-maximo-arielari

```
#include <iostream>
using namespace std;
int a;
int *p;
void memoria_local();
int main() {
  cin.clear();
  cout << "Ingrese un numero: \n";
  cin>>a;
  cout<<"\n Valor :"<<a;
  cout<<"\n Su direccion de memoria es: "<<&a;
  memoria_local();
  cout<<"\n Direccion de memoria de Puntero: "<<p;
  cout<<"\n El valor apuntado :"<<*p;
  memoria_local();
  cout<<"\n El valor apuntado :"<<*p;
  memoria_local();
  cout<<"\n El valor apuntado :"<<*p;
  memoria_local();
  cout<<"\n El valor apuntado :"<<*p;
  memoria_local();
  cout<<"\n El valor apuntado :"<<*p;

}
void memoria_local(){
  int a;
  cout<<"\n*********************************";
  cout <<"\n La memoria contiene: "<<a;
  cout <<"\n Ingrese un numero Local:";
  cin>>a;
  cout<<"\n Valor :"<<a;
  cout<<"\n Su direccion de memoria es: "<<&a;
  p=&a;
}
```

Ejemplo:

Si nos solicitan almacenar la información del dueño de una ferretería. En donde los datos a soportar son:

- Nombre
- DNI
- Dirección
- Email
- Teléfono

```
using namespace std;
const int TAM=40;
char nombre[TAM];
char email[TAM];
char direccion[TAM];
long int telefono;
int dni;
```

Test:
https://replit.com/join/hzarrahfgf-maximo-arielari

Ejemplo 2:

Si nos solicitan almacenar la información del dueño y dos de sus socios de una ferretería. En donde los datos a soportar son:

- Nombre
- DNI
- Dirección
- Email
- Teléfono

TEST:

https://replit.com/join/zbccmhqgjq-maximo-arielari

```
const int TAM=40;
//*****************Dueño*************
char due_nom[TAM];
char due_email[TAM];
char due_direccion[TAM];
long int due_telefono;
int due_dni;
//****************Socio1**************
char soc1_nom[TAM];
char soc1_email[TAM];
char soc1_direccion[TAM];
long int soc1_telefono;
int soc1_dni;
//****************Socio2**************
char soc2_nom[TAM];
char soc2_email[TAM];
char soc2_direccion[TAM];
long int soc2_telefono;
int soc2_dni;
//************************************
```

Estructuras

Struct es una palabra reservada para constituir estructuras compuestas que emplean variables primitivas u otras estructuras compuestas para generar la nueva colección de información.

```
struct  nombreDeEstructura{
                                variables primitivas;
} nombreDeVariable;
```

STRUCT

Una estructura de datos es un grupo de elementos de datos agrupados bajo un mismo nombre. Estos elementos de datos, conocidos como miembros , pueden tener diferentes tipos y diferentes longitudes. Las estructuras de datos se pueden declarar en C++ usando la siguiente sintaxis: Donde es un nombre para el tipo de estructura, puede ser un conjunto de identificadores válidos para objetos que tienen el tipo de esta estructura. Entre llaves , hay una lista con los miembros de datos, cada uno se especifica con un tipo y un identificador válido como su nombre.

Ejemplo 3:

Si nos solicitan almacenar la información del dueño y dos de sus socios de una ferretería. En donde los datos a soportar son:

- Nombre
- DNI
- Dirección
- Email
- Teléfono

```
const int TAM=40;

struct datos{
char nom[TAM];
char email[TAM];
long int telefono;
int dni;
};
//*****************Dueño*************
struct datos dueño;
//***************Socio1**************
struct datos socio1;
//***************Socio2**************
struct datos socio2;
```

TEST: https://replit.com/join/tbnhsmyglc-maximo-arielari

Inicializar Valores

Ejemplo:

```
struct info_libro{
        char titulo[60];
        char autor[30];
       char editorial[30];
       int anyo;
} libro1={“Alicia en el Pais”, “Lewis Carroll”,”Macmillan Publishers”,1865};
```

Las estructuras anidadas

Recordemos que dentro de una estructura podemos declarar cualquier tipo de dato como miembro, incluso un tipo de dato hecho por el usuario con una estructura o clase. Ahora veremos cómo hacer eso y por qué.

Ejemplo 4:

Si nos solicitan almacenar la información del dueño y dos de sus socios de una ferretería. En donde los datos a soportar son:

- Nombre
- DNI
- Dirección
- Email
- Teléfono

TEST: https://replit.com/join/fxssslpfww-maximo-arielari

```
const int TAM=40;

struct datos{
char nom[TAM];
char email[TAM];
long int telefono;
int dni;
};
struct empresa{
  datos dueño, socio1,
socio2;
} ferreteria;
```

Incluir Funciones

Es posible en el ámbito de definición incluir funciones que permitan manipular las variables que le pertenecen a la estructura.

```
struct tipo_estructura{
   tipos_de_variables;
   funciones asociadas a la estructura;
} variable_estructura;
```

Ejemplo 5:

Si nos solicitan almacenar la información del dueño y dos de sus socios de una ferretería. En donde los datos a soportar son:

- Nombre
- DNI
- Dirección
- Email
- Teléfono

TEST: https://replit.com/join/jqcgeztffx-maximo-arielari

Anexo

Ordenamiento por burbuja

Ordenamiento Burbuja

Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiandolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas «burbujas». También es conocido como el método del intercambio directo.

Ordenamiento Burbuja

```
procedimiento DeLaBurbuja (a_0, a_1, a_2, ..., a_(n-1))
    para i ← 0 hasta n − 2 hacer
        para j ← 0 hasta n − i − 2 hacer
            si a_(j) > a_(j+1) entonces
                aux ← a_(j)
                a_(j) ← a_(j+1)
                a_(j+1) ← aux
            fin si
        fin para
    fin para
fin procedimiento
```

Seudo código

Ordenamiento Burbuja

https://scratch.mit.edu/projects/846669675

Bibliografía

https://cplusplus.com/doc/tutorial/structures/

https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja