Indíce

Temario

1. Revisión de un código de estadística descriptiva
   - Media
   - Mediana
   - Moda
2. Tipos compuestos

- Estructuras

1. Apuntadores
   - Definiciones
   - Acceso a valores
   - Apuntadores y funciones

Estadística descriptiva

- Revisa el código que se encuentra en la plataforma.
- Ejemplo_Medias.c

Estructuras

Combina elementos de tipos diferentes.

```
1 struct persona {
2   unsigned int edad ;
3   unsigned int altura ;
4   char nombre [256] ;
5 } tonio , elba ;
```

Sus diferentes elementos se acceden por el operador . :

```
1 unsigned int ed = elba.edad ;
```

Definición simple

Un apuntador es un tipo que refiere a un dato en memoria; incluye :

1. La direccion en la memoria en que esta el dato considerado.
2. La manera de leer los datos a partir de este lugar de la memoria, i.e. que tipo de datos son.

Se utiliza con

```
1  tipo *var
```

---

donde “tipo" puede ser cualquier otro tipo, incluyendo apuntador!

Definición simple

Un apuntador inicializado involucra dos variables

- la variable-apuntador (cuyo valor representa una direccion en memoria),
- la variable apuntada.

Ambas variables tienen su propia direccion en memoria, y su propio tipo.

Acceso al valor

Se hace por el operador unario de indireccion *

```
1 int a = 40 ;
2 int  *a_ptr = &a ;
3 *a_p t r = a*a ;
```

Este operador tiene un nivel de prioridad 2

```
1 int a = 10 ;
2 int *b = &a ;
3 int c = a**b**b ;
4 printf ( "c = %d \n" , c ) ;
```

Acceso al valor

- Una forma equivalente al operador de indireccion es con el operador corchete

```
1 int  a  =  40  ;
2 int   *a_ptr  =  &a  ;
3 a_ptr[]  =  &a  ;
```

Acceso al valor

—

- En caso de estructuras, existe un operador sinonimo de la combinacion de * y de . : (aceso al elemento):

```
1 persona nery;
2 persona  *neryPtr = &nery ;
3 neryPtr->edad = 25 ;
4 (*neryPtr).edad = 25 ;
```

Las dos ultimas lineas son equivalentes.

Aritmetica sobre apuntadores

Un apuntador es nada mas que un entero; entonces, se puede aplicar unos operadores aritmeticos, los que sí tienen sentido :

- Sumar un entero; el resultado es un apuntador de mismo tipo, en un lugar trasladado de la memoria;
- Restar un entero; el resultado es un apuntador de mismo tipo, en un lugar trasladado de la memoria;
- Restar dos apuntadores; regresa un entero (espacio memoria entre los dos apuntadores).

Tambien se puede usar operadores de comparacion, con la condicion que se comparen apuntadores de mismo tipo.
Todas esas operaciones no son en bytes sino <en unidades correspondiendo al tipo del apuntador!

Aritmetica sobre apuntadores

```
1 int i = 15 ;
2 int *p1 , *p2 ;
3 p1 = &i ;
4 p2 = p1 + 2 ;
5 printf ("p1 = %ld\t p2 = %ld\n" , p1 , p2 ) ;
```

Aritmetica sobre apuntadores

```
1  #define N 5
2  int tab [N] = {1 , 2 , 6 , 0 , 7 } ;
3  int main ( )
4  {
5     int *p ;
6     for ( p = &tab [ 0 ] ; p <= &tab [N-1] ; p++)
7        printf ( " %d \n" , *p ) ;
8     for ( p = &tab [N-1] ; p >= &tab [ 0 ]; p--) ;
9        printf ( " %d \n" , *p ) ;
10    return 0 ;
11 }
```

Apuntadores : el operador []

- El operador [] es equivalente a una combinacion entre el operador aritmetico '+' y el operador de indireccion :

```
1 int tab [10] ;
2 int *p=&tab[0] ;
3 p[2] = 5 ;
4 *(p+2) = 5 ;
```

Las últimas dos líneas son equivalentes

Apuntadores y funciones

- Para estructuras grandes, puede ser muy costoso pasar variables de este tipo a funciones, como parametros (hay copias. . .), especialmente si usamos esta funcion muy seguido : en cambio, pasarle la direccion de esta estructura y su tipo (i.e. un apuntador) es suciente.

Apuntadores y funciones

```
1  typedef struct dataStruct {
2    double data [ 100 ] ;
3    int width ;
4    int height ;
5  } data ;
6
7  void func1 ( data mydata ) {
8  int w = mydata.width ;
9  . . .
10 } ;
11 void func2 ( data *mydata ) {
12 int w = mydata->width ;
13 . . .
14 } ;
```