Clase

Jueves8 de mayo 2014

Notas de programación del lenguaje C

Funciones

```
Tipo_retorno    nombre_Funcion(
   lista_Parametros)
{
   Cuerpo de la función
   return  expresión      /* valor que devuelve la
   función a donde se llamo */
}
```

clase

- Todos los parámetros en C se pasan por valor
- C no tiene parámetros por referencia , es necesario implementarlos con: apuntadores y el operador &

Notas de programación del lenguaje C

- Si no se declara el tipo de retorno de la función el compilador asume que el tipo de dato que devuelve es un entero
- *Resultados de una función(se le pueden mandar todos los parámetros que se quieran o pero solo regresa un solo valor)*

Una función devuelve un único valor.

El resultado se muestra con una sentencia *return*

Sintaxis *return (expresión);*

return expresion;

return;

Notas de programación del lenguaje C

- El valor devuelto(expresión) puede ser cualquier tipo de dato conocido(tipo simple o tipo estructurado)
- Se pueden regresar valores múltiples devolviendo un apuntador a una estructura o un arreglo
- El valor de retorno se aplican las mismas reglas que se aplican a un operador de asignación
- nota en el caso: si se devuelve un *int* y el tipo de retorno es un *float*, se realiza la conversión automáticamente

Notas de programación del lenguaje C

- *return* devuelve el control a la sentencia donde se llamo.
- Aunque no es obligatoria el uso de la sentencia *return en la ultima línea , se recomienda su uso.*

Notas de programación del lenguaje C

Llamadas a una función

Las funciones para ser ejecutadas , deben de ser llamadas o invocadas

Cualquier expresión puede contener una llamada a una función (redirigirá el control del programa a la función llamada)

Comúnmente las llamadas a las funciones se realizan desde la función principal main ()

Nota: también pueden ser llamadas desde otra función

Notas de programación del lenguaje C

Nota: no se puede definir una función dentro otra. Todo el código de la función debe estar listado secuencialmente. Antes de que aparezca el código de una función debe aparecer la llave de cierre de la función anterior.

Notas de programación del lenguaje C

```
La función max regresa el número mayor de dos enteros
# include <stdio.h>
int max (int x, int y)
main()
{
Int m, n;
do{
                scanf("%d %d", &m,&n);
                printf("Maximo de   %d, %d es%d\n", max(m,n));      /*llama a max  */
           }
           While( m ! = 0 );
   }
int max (int x, int y)
{
      if( x < y)
                return y;
      else
                return x ;
}
```

Notas de programación del lenguaje C

- C recomienda que se declare una función antes de llamarla

Nota:

Si una función no devuelve un resultado , se utiliza el tipo void ,que se considera como un tipo de dato especial.

Una función que no devuelve resultados , a veces se denomina procedimiento, para indicar al compilador que una función no devuelve resultado , se utiliza el tipo de retorno void

Notas de programación del lenguaje C

Direcciones de memoria

Cuando una variable se declara se asocian tres atributos: su nombre , su *tipo* y su *dirección* de *memoria*

int pato

Ox4ff456

pato | 45 |

int

Notas de programación del lenguaje C

Asocia al nombre pato , el tipo entero y la dirección de la memoria donde se almacena el valor de pato

La caja de este ejemplo representa la posición de almacenamiento de la memoria , el nombre de la variable esta a la izquierda de la caja , la dirección de la variable esta arriba de la caja y el tipo esta debajo de la caja , y el valor de la variable esta dentro de la caja

Notas de programación del lenguaje C

- Al valor de la variable se accede por medio de su nombre

  printf(“%d”, pato);
- A la dirección de la variable se accede mediante el operador dirección **&**

  printf(“%p”, &pato);

El operador de dirección “&” <<opera>>(se aplica) sobre el nombre de la variable para obtener sus direcciones

Notas de programación del lenguaje C

Apuntadores

Al declara una variable en C, el compilador establece una área de memoria para almacenar el contenido de la variable.

Cuando se declara una variable de int, el compilador asigna dos bytes de memoria , este espacio se sitúa en una posición especifica de la memoria conocida como *dirección de memoria*

Notas de programación del lenguaje C

Cuando se referencia (se hace uso) al valor de la variable, el compilador accede automáticamente a la dirección de la memoria donde se almacenada la variable.

Se puede ganar eficiencia en el acceso a esta dirección de memoria utilizando apuntadores

Notas de programación del lenguaje C

Obtener el valor y la dirección de una variable

```
#include  <stdio.h>
main ( )
{
    int oso =  39;
    printf(“ oso= %d\n”, oso);      /*muestra el valor de la variable oso */
printf(“ oso= %p\n”, &oso);             /*muestra el valor de la dirección
    memoria  donde se encuentra el valor  de la variable oso */
}
```

Oso =345   &oso = Ox45ff3dd

Notas de programación del lenguaje C

- Una variable que se declara en C tiene una dirección asociada a ella
- Un apuntador es una dirección de memoria

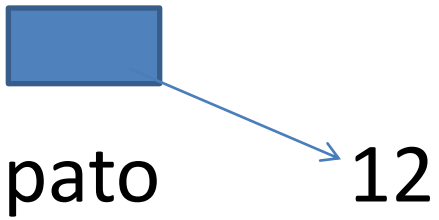
Nota: un apuntador indica donde encontrar algo, un apuntador es la *dirección de una variable*

Notas de programación del lenguaje C

Reglas de los apuntadores:

- Es una variable como cualquier otra
- La variable apuntador contiene una dirección que apunta a una posición de memoria
- En esa posición se almacena los datos a los que apunta el apuntador
- Un apuntador apunta a una variable en memoria

Notas de programación del lenguaje C



El valor de un apuntador es una dirección de memoria, la dirección depende del estado de la computadora donde se ejecuta el programa

Nota:

- Al tipo de variable que almacena una dirección se denomina apuntador

Notas de programación del lenguaje C

```
#include  <stdio.h>
main ( )
{
    int oso =  39; int *perro = &oso;
    printf(“ oso= %d, &oso  = %p, perro =%p \n”, oso, &oso, perro);      /*perro
    variable apuntador, tiene la dirección de oso */
printf(“ &perro= %p\n”, &perro);
}
```

oso =345   &oso = Ox45ff3dd perro= Ox45ff3dd   &perro = Ox4fffd30

Notas de programación del lenguaje C

- oso =345 &oso = Ox45ff3dd perro= Ox45ff3dd &perro = Ox4fffd30

  Ox4fffd30 Ox45ff3dd

int * int

La variable perro se denomina ***apuntador*** esto porque ***apunta*** a la posicion de otro valor

Es un apuntador ***int*** cuando el valor al que apunta es de tipo ***int***

Notas de programación del lenguaje C

- Se declaran como cualquier variable, y deben de ser declaradas antes de utilizarse.
- La declaración de una variable apuntador debe indicar al compilador el tipo de dato al que apunta el apuntador ; se debe colocar un asterisco (*)antes del nombre de la variable

sintaxis

<tipo de dato apuntado> *<identificador del apuntador>

Notas de programación del lenguaje C

- Un operador * en una declaración indica que la variable declarada almacenará una dirección de un tipo de datos especificado
- la inicialización de un apuntador proporciona a ese apuntador la dirección del dato , se puede utilizar el apuntador para referenciar los datos direccionados
- Para asignar una dirección de memoria a un apuntador es necesario el operador de referencia ***&***

  &oso significa la dirección de oso

El ***operador &*** devuelve la dirección de la variable a la que se aplica

- Ejemplos

Iniciación de apuntadores

Asigna memoria(estática) definiendo una variable y aquí el apuntador apunta al valor de la variable

```
int sapo;        /*define una variable sapo*/
Int *gato;       /*define un apuntador a un entero gato*/
gato = &sapo  /* asigna la dirección de sapo a gato */
```

Notas de programación del lenguaje C

- apuntadores y arreglos

Los arreglos (*arrays*) y apuntadores están fuerte relacionados. Se pueden direccionar arreglos como si fueran apuntadores y apuntadores como si fueran arreglos

La posibilidad de acceder y almacenar a un apuntador y un arreglos, implica que se pueden almacenar cadenas de datos en elementos de arreglos. Sin apuntadores eso no es posible, ya que no existe el tipo de dato cadena (string) en C, no existen variables de cadena, únicamente constantes de cadena

Arreglos

Notas de estructura de datos con lenguaje C

Arreglos

Los *arreglos* son estructuras de datos donde variables del mismo tipo son llamadas bajo un mismo nombre pero se distinguen por un índice.

Contienen un número determinado de elementos (su tamaño) y todos los elementos han de ser del mismo tipo de datos; es una estructura de datos homogénea

Esta característica supone una gran limitación cuando se requieren grupos de elementos con tipos diferentes de datos cada uno

Notas de estructura de datos con lenguaje C

- Por ejemplo, si se dispone de una lista de temperaturas, es útil un *arreglos*;
- para una lista de información de clientes que contenga elementos tales como nombre, la edad, la dirección, el número de la cuenta, etc., los *arreglos* no son adecuados. La solución a este problema es utilizando un ***tipo de dato registro, estructura*** denominado en lenguaje C.

Notas de estructura de datos con lenguaje C

- Los componentes individuales de una estructura se llaman *miembros*.
- Cada miembro (elemento) de una estructura puede contener valores de un tipo diferente de dato
- Se puede utilizar una estructura para almacenar diferentes tipos de información sobre una persona, tal como nombre, estado civil, edad y fecha de nacimiento. Cada uno de estos elementos se denomina *nombre del miembro*.

Notas de estructura de datos con lenguaje C

- **Una estructura es una colección de un tipo o más tipos de elementos denominados miembros, cada uno de los cuales puede ser un tipo de dato diferente.**
- Una estructura puede contener cualquier número de miembros, cada uno de los cuales tiene un nombre único, denominado *nombre* del miembro

Notas de estructura de datos con lenguaje C

- Ejemplo: almacenar los datos de la venta de libros(**lib**) de matemáticas. Estos datos pueden ser:
- Título
- Autor
- Número de volúmenes
- Precio
- Fecha de compra

Notas de estructura de datos con lenguaje C

- La estructura lib contiene cualquier número de miembros. Tras decidir los miembros, se debe decidir cuáles son los tipos de datos a utilizar por los miembros:

Nombre de Miembro	Tipo de dato
Título	*Arreglo* de caracteres de tamaño 30.
Autor	*Arreglo* de caracteres de tamaño 25.
Número de volúmenes	Entero
Precio	Punto flotante
Fecha de compra	*Arreglo* de caracteres de tamaño 8.

Notas de estructura de datos con lenguaje C

Declaración de una estructura

- Una estructura es un tipo de dato definido por el programador, este se debe declarar antes de utilizarlo. El formato de la declaración es:

struct <nombre de la estructura>

{

  <tipo de dato miembro$_1$> <nombre miembro$_1$>

  <tipo de dato miembro$_2$> <nombre miembro$_2$>

…

  <tipo de dato miembro$_n$> <nombre miembro$_n$>

};

Notas de estructura de datos con lenguaje C

La declaración de la estructura lib es:

```
struct lib
{
  char titulo [30];
  char autor [25];
  int num_volumenes;
  float precio;
  char fecha_compra [8];
};
```

Notas de estructura de datos con lenguaje C

```
struct venta
{
  char vendedor [30];
  unsigned int codigo;
  int unids_articulo;
  float precio_unit;
};
```

Notas de estructura de datos con lenguaje C

Definición de variables de estructuras

- Al igual que a los tipos de datos enumerados, a esta estructura se accede utilizando una variable o variables que se deben definir después de la declaración de la estructura.
- En C existen dos conceptos similares a considerar, *declaración y definición.*

Notas de estructura de datos con lenguaje C

La diferencia técnica es la siguiente:

- ***una declaración*** especifica simplemente el nombre y el formato de la estructura de datos, pero no reserva almacenamiento en memoria;
- la declaración especifica un nuevo tipo de dato: struct <nombre_estructura>.
- Sin embargo, cada definición de variable para una estructura dada crea un área en memoria en donde los datos se almacenan de acuerdo al formato estructurado declarado.

Notas de estructura de datos con lenguaje C

Las variables de estructuras se pueden definir de dos formas:

1) Listándolas inmediatamente después de la llave de cierre de la declaración de la estructura

2) Listando el tipo de la estructura creado seguido por las variables correspondientes en cualquier lugar del programa antes de utilizarlas.

Notas de estructura de datos con lenguaje C

Forma1

```
struct lib
{
  char titulo [30];
  char autor [25];
  int num_volumenes;
  float precio;
  char fecha_compra [8];
}; matematicas1, matematicas2, matematicas3;
```

forma2

```
struct lib matematicas1, matematicas2, matematicas3;
```

Notas de estructura de datos con lenguaje C

ejemplo de definición/declaración

programa que gestiona libros y procese los siguientes datos: título del libro, nombre del autor, editorial y año de publicación. La estructura info_libro :

```
struct info_libro
{
   char titulo [60];
   char autor [30];
   char editorial [30];
   int anyo;
}; libro1, libro2, libro3;
```

Forma 2

```
struct info_libro libro1, libro2, libro3;
```

Notas de estructura de datos con lenguaje C

Uso de estructuras en asignaciones

Como una estructura es un tipo de dato similar a un int o un char, se puede asignar una estructura a otra. Por ejemplo, se puede hacer que libro3, libro4 y libro5 tengan los mismos valores en sus miembros que libro1. Por consiguiente, sería necesario realizar las siguientes sentencias:

```
libro3 =libro1;
libro4 =libro1;
libro5 =libro1;
```

De modo alternativo se puede escribir

```
libro5 =libro4= libro3 =libro1;
```

Notas de estructura de datos con lenguaje C

Inicialización de una declaración de estructuras

Se puede inicializar una estructura de dos formas:

Dentro de la sección de código de un programa, o bien se puede inicializar la estructura como parte de la definición.

Cuando se inicializa una estructura como parte de la definición, se especifican los valores iniciales, entre llaves, después de la definición de variables estructura. El formato general es este caso:

struct <tipo> <nombre variable estructura> =

{valor miembro$_1$,

valor miembro$_2$,

…

valor miembro$_n$,

};

Notas de estructura de datos con lenguaje C

Acceso a una estructura de datos mediante el operador puntero

- El operador apuntador **->** sirve para acceder a los datos de la estructura a partir de un apuntador.
- Para utilizar este operador se debe definir primero una variable apuntador para apuntar a la estructura

Notas de estructura de datos con lenguaje C

- el operador apuntador se utiliza para apuntar a un miembro dado.
- La asignación de datos a estructuras utilizando el operador apuntador tiene el formato:

  <apuntador estructura> **->** <nombre miembro> = datos;

Notas de estructura de datos con lenguaje C

por ejemplo, una estructura estudiante:

```
    struct estudiante
    {
       char Nombre[41];
       int Num_Estudiante;
       int Anyo_de_matricula;
       float Nota;
};
```

Notas de estructura de datos con lenguaje C

Se puede definir ptr_est como puntero a la estructura:

struct estudiante *ptr_est;

struct estudiante mejor;

- A los miembros de la estructura estudiante se pueden asignar datos como sigue (siempre y cuando la estructura ya tenga creado su espacio de almacenamiento por ejemplo, con malloc(); o bien, tenga la dirección de una variable estructura).

ptr_est=&mejor; /*ptr_est tiene la dirección (apunta a) mejor */

Notas de estructura de datos con lenguaje C

```
ptr_est=&mejor;          /*ptr_est tiene la
   dirección (apunta a) mejor */

strcpy (ptr_est ->Nombre, “Francisco Ruiz”);
ptr_est –> Num_Estudiante =20113425;
   ptr_est -> Nota = 9.98;
```

Notas de estructura de datos con lenguaje C

- Nota: **Consejo de programación**
- Antes de acceder a los miembros de una estructura con una variables apuntador y el operador **->**, es preciso crear espacio de almacenamiento en memoria; por ejemplo, con la función malloc ( ).

Notas de estructura de datos con lenguaje C

Lectura de información de una estructura

Para darle datos (introducir la información) en la estructura basta con acceder a los miembros de la estructura con el operador punto (“.”) o flecha (apuntador ->).

Se puede introducir la información desde el teclado o desde un archivo, o asignar valores calculados.

Notas de estructura de datos con lenguaje C

Si z es una variable de tipo estructura complejo, se lee parte real, parte imaginaria y se calcula el módulo:

```
struct complejo
{
  float pr;
  float pi;
  float modulo;
};
struct complejo z; printf("\n Parte real: ");
scanf ("%f", &z.pr);
printf("\nParte imaginaria: ");
scanf ("%f", &z.pi);
/*calculo del módulo*/
z.modulo = sqrt(z.pr*z.pr + z.pi*z.pi);
```

Notas de estructura de datos con lenguaje C

Recuperación de información de una estructura

- Se recupera información de una estructura utilizando el operador de asignación o una sentencia de salida (printf(), puts(), …).
- Se puede emplear el operador punto o el operador flecha (apuntador)

Notas de estructura de datos con lenguaje C

formato general tiene estas dos formas:

<nombre variable> = <nombre variable estructura>.<nombre miembro>;

O bien

<nombre variable> = <puntero de estructura> -> <nombre miembro>;

Para salida:

printf(“ ”, <nombre variable estructura>.<nombre miembro>);

o bien

printf(“ ”, <puntero de estructura>-> <nombre miembro>);

Notas de estructura de datos con lenguaje C

uso de la estructura complejo:

```
float x, y;
struct complejo z;
struct complejo *pz;
pz =&z; x =z.pr; y = z.pi;
…
printf (“\n Númerp complejo (%.1f, %.1f),
   módulo: %.2f”, pz->pr, pz->pi, pz-> modulo);
```

Notas de estructura de datos con lenguaje C

SINÓNIMO DE UN TIPO DE DATOS: typedef

La sentencia **typedef** permite al crear un sinónimo de un tipo de dato definido por el programador o de un tipo ya existente

typedef para declarar un nuevo nombre, gato, de tipo de dato double

typedef double gato;

gato, de tipo dato, en este ejemplo sinónimo de double

Notas de estructura de datos con lenguaje C

- Si definimos una función Distancia ( ) de tipo gato:

```
gato Distancia (const Punto& p, const Punto&
   p2)
{
   …
   gato longitud = sqrt (r-cua);
   return longitud;
}
```

Notas de estructura de datos con lenguaje C

Sintaxis

typedef tipo_dato_definido nuevo_nombre;

- Puede declararse un tipo estructura o un tipo unión y a continuación asociar el tipo estructura a un nombre con typedef.

Notas de estructura de datos con lenguaje C

Ejemplo: declaración del tipo de dato complejo y asociación a complex.

```
struct complejo
{
  float x, y;
};
typedef struct complejo complex;
/* definición de un array de complejos*/
complex v[12];
```

Notas de estructura de datos con lenguaje C

Declaración del tipo de dato racional y asociación a Racional.

```
typedef struct racional
{
   int numerador;
   int denominador;
} Racional;
```

Notas de estructura de datos con lenguaje C

Ahora se puede declarar la estructura número utilizando el tipo complex y el tipo Racional:

```
struct numero
{
  complex a;
  racional r;
};
```

- La ventaja de typedef es que permite dar nombres de tipos de datos más acordes con aquello que representan en una determinada aplicación.

Notas de estructura de datos con lenguaje C

ESTRUCTURAS ANIDADAS

- Una estructura puede contener otras estructuras por esta razón les llamamos *estructuras anidadas*.
- Las estructuras anidadas ahorran tiempo en la escritura de programas que utilizan estructuras similares.
- Se deben definir los miembros comunes sólo una vez en su propia estructura
- y a continuación utilizar esa estructura como un miembro de otra estructura

Notas de estructura de datos con lenguaje C

- Ejemplo

```
struct empleado
{
  char nombre_emp [30];
  char direccion [25];
  char ciudad [20];
  char provincia [20];
  long int cod_postal;
  double salario;
};
```

Notas de estructura de datos con lenguaje C

```
struct clientes
{
  char nombre_cliente [30];
  char direccion [25];
  char ciudad [20];
  char provincia [20];
  long int cod_postal;
  double saldo;
};
```

- Estas estructuras contienen muchos datos diferentes, aunque hay datos que están ocultos.

Notas de estructura de datos con lenguaje C

- se podría disponer de una estructura, info_dir, que contuviera los miembros comunes.

```
struct info_dir
{
  char direccion [25];
  char ciudad [20];
  char provincia [20];
  long int cod_postal;
};
```

Notas de estructura de datos con lenguaje C

- Una estructura se puede utilizar como un miembro de las otras estructuras, es decir, *anidarse.*

```
struct empleado
{
   char nombre_emp [30];
   struct info_dir direccion_emp;
   double salario;
};
```

Notas de estructura de datos con lenguaje C

```
struct cliente
{
   char nombre_cliente [30];
   struct info_dir direccion_clien;
   double saldo;
};
```

Notas de estructura de datos con lenguaje C

- ejemplo

Notas de estructura de datos con lenguaje C

- **ARREGLOS (ARRAYS) DE ESTRUCTURA**

Se puede crear un *arreglo* de estructuras tal como se crea un *arreglo* de otros tipos.

Los **arreglos de estructuras** son idóneos para almacenar un archivo completo de empleados, un archivo de inventario, o cualquier otro conjunto de datos que se adapte a un formato de estructura

Mientras que los ***arreglos*** proporcionan un medio practico de almacenar diversos valores del mismo tipo

los **arreglos de estructuras** le permiten almacenar juntos diversos valores de diferentes tipos, agrupados como estructuras.

Notas de estructura de datos con lenguaje C

- Por ejemplo la declaración de un *arreglo* de estructuras info_libro se puede hacer de un modo similar a cualquier *arreglo*, es decir,

  struct info_libro libros [100];

asigna un array de 100 elementos denominados libros

Para acceder a los miembros de cada uno de los elementos estructura se utiliza una notación de *arreglo*

Notas de estructura de datos con lenguaje C

Para inicializar el primer elemento de libros, su código debe hacer referencia a los miembros de libros [0] de la forma siguiente:

strcpy (libros[0]. titulo, “Historia de México”);

strcpy (libros[0]. autor, “Luis Moya”);

strcpy (libros[0]. editorial, “McGraw-Hill”);

libros[0].anyo=1999;

Notas de estructura de datos con lenguaje C

- También puede inicializarse un *arreglo* de estructuras en el punto de la declaración encerrando la lista de inicializadores entre llaves, {}. Por ejemplo,

struct info_libros libros[3]={“Historia de México”, “Luis Moya”, “McGraw-Hill”, 1999, “Los Aztecas”, “Jorge García López”, “McGraw-Hill”, 1999, “El Mundo Maya”, “Rosendo Márquez”, “McGraw-Hill”, 1997};

Notas de estructura de datos con lenguaje C

El siguiente ejemplo se declara una estructura que representa a un número racional, un *arreglo* de números racionales es inicializado con valores al azar.

```
struct racional
{
   int N,
   int D;
};
struct racional rs[4]={1, 2, 2, 3, -4, 7, 0, 1};
```

Tipos de datos abstractos (TDA)

Notas de estructura de datos con lenguaje C

- La modularidad es la posibilidad de dividir una aplicación en piezas más pequeñas llamadas módulos.
- Abstracción de datos es la técnica de inventar nuevos tipos de datos que sean más adecuados a una aplicación y por consiguiente, facilitar la escritura del programa. La técnica de abstracción de datos en una técnica potente de propósito general que cuando se utiliza adecuadamente, puede producir programas más cortos, más legibles y flexibles.
- Los objetos combinan en una sola unidad datos y funciones que operan sobre esos datos.

Notas de estructura de datos con lenguaje C

- Los lenguajes de programación soportan en sus compiladores tipos de datos fundamentales o básicos (predefinidos), por ejemplo: int, char y float en C
- La mayoría de los lenguajes de programación tienen características que permiten ampliar el lenguaje añadiendo sus propios tipos de datos.

Notas de estructura de datos con lenguaje C

- Un tipo de dato definido por el programador se denomina tipo abstracto de dato, **TAD**, (abstract data type **ADT**).
- El término abstracto se refiere al medio en que un programador abstrae algunos conceptos de programación creando un nuevo tipo de dato.

Notas de estructura de datos con lenguaje C

- La modularización de un programa utiliza la noción de tipo abstracto de dato (**TAD**) siempre que sea posible.
- Si el lenguaje de programación soporta los tipos que desea el programador y el conjunto de operaciones sobre cada tipo, se obtienen un nuevo tipo de dato denominado TAD.

Notas de estructura de datos con lenguaje C

Conceptos Claves	
•Abstracción •Clase •Componentes •Encapsulación •Interfaz •Modularidad	•Objeto •Rol •Reutilización •Software •Tipos de datos y variables

Notas de estructura de datos con lenguaje C

EL PAPEL DE LA ABSTRACCIÓN

el problema de la complejidad de la programación de la informática

- Se han desarrollado mecanismos utilizados por los programadores para controlar la complejidad como:

Se destaca ***la abstracción***

Notas de estructura de datos con lenguaje C

- Como describe Wulft “los humanos hemos desarrollado una técnica excepcionalmente potente para tratar la complejidad: abstraernos de ella. Incapaces de dominar en su totalidad los objetos complejos, se ignoran los detalles no esenciales, tratando en su lugar con el modelo ideal del objeto y concentrándonos en el estudio de sus aspectos esenciales”.

Notas de estructura de datos con lenguaje C

La abstracción es la capacidad para encapsular y aislar la información, del diseño y ejecución.

La progresión histórica de la abstracción en la programación comienza en los procedimientos y sigue en los módulos, tipos abstractos de datos y objetos.

Notas de estructura de datos con lenguaje C

La abstracción como un proceso natural mental

Las personas normalmente comprenden el mundo construyendo modelos mentales de partes del mismo, tratan de comprender cosas con las que pueden interactuar:

Las personas normalmente comprenden el mundo construyendo modelos mentales de partes del mismo, tratan de comprender cosas con las que pueden interactuar:

Notas de estructura de datos con lenguaje C

- El proceso de construcción de modelos es lo mismo que el diseño de *software*
- El desarrollo de *software* es único, el diseño de *software* produce el modelo que puede ser ejecutado por una computadora.
- Los modelos mentales deben ser más sencillos que el sistema al cual imitan, o en caso contrario serán inútiles

Notas de estructura de datos con lenguaje C

- Ejemplo:

Consideremos un mapa como modelo de su territorio.

el mapa debe ser más sencillo que el territorio que modela.

Que nos proporciona un mapa nos abstrae sólo aquellas características del territorio que deseamos modelar

Notas de estructura de datos con lenguaje C

- Un mapa de carreteras modela cómo llegar de una posición a otra
- Los modelos mentales abstraen esas características de un sistema requerido para su comprensión

  mientras ignoran características irrelevantes

Este proceso de *abstracción* es psicológicamente necesario y natural, la abstracción es crucial para comprender este complejo mundo.

Notas de estructura de datos con lenguaje C

De la historia: a principio de la computación, los programadores enviaban instrucciones binarias a una computadora manipulando directamente interrupciones en sus paneles frontales. Los nemotécnicos del lenguaje ensamblador eran abstracciones diseñadas para evitar que los programadores tuvieran que recordar las secuencias de bits que componen las instrucciones de un programa. El siguiente nivel de abstracción se consigue agrupando instrucciones primitivas para formar macroinstrucciones.

Notas de estructura de datos con lenguaje C

Ejemplo: un conjunto se puede definir por la abstracción como una colección no ordenada de elementos en el que no existen duplicados

De lo anterior se pueden especificar si sus elementos se almacenan en un *arreglo*, una lista enlazada o cualquier otra estructura de datos.

Notas de estructura de datos con lenguaje C

- Un conjunto de instrucciones realizadas por el programador se pueden invocar por una macroinstrucción
- Una macroinstrucción instruye a la máquina para que realice algunas tareas
- Tras los lenguajes de programación ensambladores aparecieron los lenguajes de programación de alto nivel, que supusieron un nuevo nivel de abstracción.

Notas de estructura de datos con lenguaje C

Los lenguajes de programación de alto nivel permitieron a los programadores alejarse de las interioridades arquitectónicas específicas de una máquina dada.

Cada instrucción en un lenguaje de alto nivel puede invocar varias instrucciones máquina, dependiendo de la computadora donde se compila el programa.

Esta abstracción permitía a los programadores escribir *software* para propósito genérico, sin preocuparse sobre que máquina corría el programa.

Notas de estructura de datos con lenguaje C

- Secuencias de sentencias de lenguajes de alto nivel se pueden agrupar en procedimientos y se invocan por una sentencia.
- La programación estructurada alienta el uso de abstracciones de control tales como bucles o sentencias if-then que se han incorporado en lenguajes de alto nivel
- sentencias de control permitieron a los programadores abstraer las condiciones comunes para cambiar la secuencia de ejecución.

Notas de estructura de datos con lenguaje C

- El proceso de abstracción fue evolucionando desde la aparición de los primeros lenguajes de programación.
- El método idóneo para controlar la complejidad de los problemas ha sido aumentar los niveles de abstracción.
- *Abstraer* un problema supone la capacidad de encapsular y aislar la información de diseño y ejecución.

Notas de estructura de datos con lenguaje C

- La progresión histórica de la abstracción va desde las estructuras de control, pasando por los procedimientos, los módulos, los tipos de datos y los objetos.

Procedimientos

Los procedimientos y funciones fueron unos de los primeros mecanismos de abstracción que se utilizaron ampliamente en lenguajes de programación.

Notas de estructura de datos con lenguaje C

- Los procedimientos permitían tareas que se ejecutaban rápidamente, o eran ejecutados sólo con ligeras variaciones, que se reunían en una entidad y se reutilizaban, en lugar se duplicar el código varias veces
- El procedimiento proporciono la primera posibilidad de *ocultación de información.*

Notas de estructura de datos con lenguaje C

- Un programador podía escribir un procedimiento o conjunto de procedimientos, que serían utilizados por otros programadores y que no necesitaban conocer los detalles de implementación, sólo necesitaban el interfaz
- los procedimientos no resolvían todos los problemas. En particular, no eran un mecanismo efectivo para ocultar la información y para resolver un problema que se producía al trabajar múltiples programadores con nombres idénticos.

Notas de estructura de datos con lenguaje C

Módulos

- Un módulo es una técnica que proporciona la posibilidad de dividir sus datos y procedimientos en una *parte privada* –sólo accesible dentro del módulo- y una *parte pública* –accesible dentro del módulo- y una *parte pública* –accesible fuera del módulo-.

Los tipos, de datos (variables) y procedimientos se pueden definir en cualquier parte.

Notas de estructura de datos con lenguaje C

- El criterio a seguir en la construcción de un módulo es que si no se necesita algún tipo de información, no se debe tener acceso a ella. Este criterio es *la ocultación de información*
- Los módulos resuelven algunos problemas, pero no todos los problemas del desarrollo de *software*.
- Los módulos proporcionan un método efectivo de ocultación de la información, pero no permiten realizar *instanciación*, que es la capacidad de hacer múltiples copias de las zonas de datos.

Notas de estructura de datos con lenguaje C

Tipos abstractos de datos

- Un *tipo abstracto de datos* (**TAD**) es un tipo de dato definido por el programador que se puede manipular de un modo similar a los tipos de datos definidos por el sistema.
- Al igual que los tipos definidos por el sistema, un *tipo de dato abstracto* corresponde a un conjunto (puede ser de tamaño indefinido) de valores legales de datos y un número de operaciones primitivas que se pueden realizar sobre esos valores.

Notas de estructura de datos con lenguaje C

- Se pueden crear variables con valores que están en el rango de valores legales y pueden operar sobre esos valores utilizando las operaciones definidas.
- Los módulos se utilizan frecuentemente como una técnica de implementación para tipos abstractos de datos, y el tipo abstracto de datos es un concepto más teórico.

Notas de estructura de datos con lenguaje C

Para construir un tipo abstracto de datos se debe poder:

1. Exponer una definición del tipo.
2. Hacer disponible un conjunto de operaciones que se puedan utilizar para manipular instancias de ese tipo.
3. Proteger los datos asociados con el tipo de modo que sólo se puede actuar sobre ellas con las rutinas proporcionadas.
4. Hacer instancias múltiples del tipo.

Los módulos son mecanismos de ocultación de información y no cumplen básicamente más que los tres primeros apartados.

Notas de estructura de datos con lenguaje C

Objetos

Un **objeto** es sencillamente un tipo abstracto de datos al que se añaden importantes innovaciones en compartición de código y reutilización.

Los mecanismos básicos de orientación a objetos son: *objetos, mensajes y métodos, clases e instancias y herencia.*

Notas de estructura de datos con lenguaje C

Una idea fundamental es la comunicación de los objetos a través de *paso de mensajes*.

Además de esta idea, se añaden los mecanismos de *herencia y polimorfismo*.

La herencia permite diferentes tipos de datos para compartir el mismo código, permitiendo una reducción en el tamaño del código y un incremento en la funcionalidad.

El polimorfismo permite que un mismo mensaje pueda actuar sobre objetos diferentes y comportarse de modo distinto.

Notas de estructura de datos con lenguaje C

La *persistencia* se refiere a la permanencia de un objeto, esto es, la cantidad de tiempo para el cual se asigna espacio y permanece accesible en la memoria del computador.

Notas de estructura de datos con lenguaje C

Notas de estructura de datos con lenguaje C

TIPOS DE DATOS

Todos los lenguajes de programación soportan algún tipo de datos

En C soporta: base como enteros, reales y caracteres; así como tipos compuestos tales como *arreglos* (vectores y matrices) y *estructuras* (registros).

Notas de estructura de datos con lenguaje C

Un valor depende de su representación y de la interpretación de la representación, por lo que una definición informal de un tipo de dato es: *Representación + Operaciones.*

Un *tipo de dato* describe un conjunto de objetos con la misma representación

Notas de estructura de datos con lenguaje C

Existen un número de operaciones asociadas con cada tipo.

Es posible realizar aritmética sobre tipos de datos enteros y reales, concatenar cadenas o recuperar o modificar el valor de un elemento.

La mayoría de los lenguajes tratan las variables y constantes de un programa como *instancias* de un *tipo de dato*

Notas de estructura de datos con lenguaje C

Un tipo de dato proporciona una descripción de sus instancias que indican al compilador cosas como cuánta memoria se debe asignar para una instancia, cómo interpretar los datos en memoria y qué operaciones son permisibles sobre esos datos

Ejemplo: cuando se escribe una declaración tal como float Pato en C ó C++, se está declarando una instancia denominada Pato del tipo de dato float

Notas de estructura de datos con lenguaje C

El tipo de datos float indica al compilador que reserve, por ejemplo, 32 bits de memoria y que operaciones tales como *“sumar” y “multiplicar”* esta permitidas, mientras que operaciones tales como el *“el resto” (módulo y “desplazamiento de bits”* no lo están

Notas de estructura de datos con lenguaje C

Los tipos de datos que se construyen en un compilador de este modo se conocen como *tipos de datos fundamentales (predefinidos),* y por ejemplo en C y C++ son entre otro: int, char, float y double.

Nota: El programador no tienen que preocuparse de saber cómo el compilador del lenguaje implementa los tipos de datos predefinidos, simplemente usa los tipos de datos en el programa.

Notas de estructura de datos con lenguaje C

Cada lenguaje de programación incorpora una colección de tipos de datos fundamentales, que incluyen normalmente enteros, reales, carácter, etc.

Los lenguajes de programación soportan también un número de constructores de tipos incorporados que permiten generar tipos más complejos.

Por ejemplo, C soporta registros (estructuras) y arreglos(*arrays)*.