Guía práctica de estudio 03.
Tipo de dato abstracto

Elaborado por:

M.C. Edgar E. García Cano

Ing. Jorge A. Solano Gálvez

Autorizado por:

M.C. Alejandro Velázquez Mena

Guía práctica de estudio 03. Tipo de dato abstracto

Objetivo:

Utilizarás estructuras en lenguaje *C* para modelar tipos de dato abstracto e implementarlos en las estructuras de datos lineales.

Actividades:

- Crear estructuras en lenguaje C.
- Crear tipos de datos utilizando estructuras.

Introducción

Un tipo de dato abstracto (TDA) es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico. Un TDA es una abstracción que permite modelar las características de un elemento en particular.

Un tipo de dato abstracto se puede manipular de forma similar a los tipos de datos que están predefinidos dentro del lenguaje de programación, encapsulando más información, según se requiera.

La implementación de un tipo de dato abstracto depende directamente del lenguaje de programación que se utilice. En lenguaje *C* los tipos de dato abstracto se crean mediante las estructuras (struct).

Licencia GPL de GNU

El software presente en esta guía práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```
/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
```

```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 * Author: Jorge A. Solano
 *
*/
```

Estructuras en lenguaje C

Una estructura es una colección de una o más variables, de iguales o diferentes tipos, agrupadas bajo un solo nombre, es decir, es un tipo de dato compuesto que permite almacenar un conjunto de datos de diferente tipo (agrupar un grupo de variables relacionadas entre sí) que pueden ser tratadas como una unidad (bajo un mismo nombre).

Las estructuras pueden contener tipos de datos simples y tipos de datos compuestos. Los tipos de datos simples (o primitivos) son: carácter, números enteros o números de punto flotante. Los tipos de datos compuestos son: los arreglos y las estructuras.

Por lo tanto, los tipos de datos abstractos (TDA) en lenguaje *C* se pueden crear a través de una estructura.

Cada ente u objeto es una abstracción de un elemento y, por ende, se puede modelar a través de una estructura en lenguaje *C*: una película, un video, una pista musical, un documento a imprimir, las armas de un juego, etc.

La sintaxis para crear estructuras en lenguaje *C* está definida por la palabra reservada *struct*, seguida del nombre de la estructura y, entre llaves, se definen el número y tipo de variables que definan al nodo (abstracción), es decir:

```
struct nodo {
        tipoDato elemento1;
        tipoDato elemento2;
        …
        tipoDato elementoN;
};
```

Para crear una variable de un tipo de dato abstracto, se debe especificar el nombre de la estructura y el nombre de la variable, es decir:

```
struct nodo elemento;
```

donde elemento es el nombre de la variable con la que se puede acceder a los datos definidos dentro de la estructura.

Código (Nodo película)

```
#include<stdio.h>

struct pelicula{
      char *nombre;
      char *genero;
      short anio;
      short numDirectores;
      char *directores[10];
};

void imprimirDatosPelicula(struct pelicula);
struct nodo llenarDatosPelicula(char *, char *, short , short , char
*[10]);
```

```
int main(){
    char *directores[10];
    directores[0] = "Lana Wachowski";
    directores[1] = "Andy Wachowski";
    struct pelicula matrix = llenarDatosPelicula("The matrix", "Ciencia
ficción", 1999, 2, directores);
    imprimirDatosPelicula(matrix);
    return 0;
}

struct nodo llenarDatosPelicula(char *nombre, char *genero, short anio,
short numDirectores, char *directores[10]){
    struct pelicula movie;
    movie.nombre = nombre;
    movie.genero = genero;
    movie.anio = anio;
    movie.numDirectores = numDirectores;
    int cont = 0;
    for ( ; cont < movie.numDirectores ; cont++){
        movie.directores[cont] = directores[cont];
    }
    return movie;
}

void imprimirDatosPelicula(struct pelicula movie){
    printf("PELICULA: %s\n", movie.nombre);
    printf("GENERO: %s\n", movie.genero);
    printf("ANIO: %d\n", movie.anio);
    printf("DIRECTOR(ES):\n");
    int cont = 0;
    for ( ; cont < movie.numDirectores ; cont++){
        printf("%s\n", movie.directores[cont]);
    }
}
```

En el ejemplo anterior se puede observar el modelado de algunas de las características que puede tener una película.

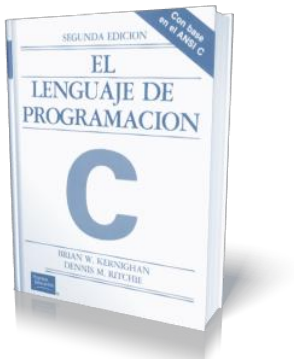
Código (Pila de películas)

```
#include<stdio.h>

#define TAM 2
#define NUM_DIR 2

struct pelicula{
      char nombre[20];
      char genero[20];
      short anio;
      short numDirectores;
      char directores[NUM_DIR][20];
};

void llenarArreglo(struct pelicula *);
void imprimirArreglo(struct pelicula *);

int main(){
    struct pelicula arreglo[TAM];
    llenarArreglo (arreglo);
    imprimirArreglo (arreglo);
    return 0;
}

void llenarArreglo(struct pelicula arreglo [TAM]){
    int iesimo, enesimo;
    for (iesimo=0 ; iesimo<TAM ; iesimo++) {
        struct pelicula movie;
        printf("####### Película %d #######\n", iesimo+1);
        printf("Ingrese nombre película:");
        setbuf(stdin, NULL);
        scanf("%s", movie.nombre);
        getchar();
        printf("Ingrese género película:");
        setbuf(stdin, NULL);
        scanf("%s", movie.genero);
        getchar();
        printf("Ingrese año película:");
        setbuf(stdin, NULL);
        scanf("%d", &movie.anio);
        movie.numDirectores = NUM_DIR;
        for (enesimo=0 ; enesimo<NUM_DIR ; enesimo++){
            printf("Ingrese director %d:", enesimo+1);
            setbuf(stdin, NULL);
            scanf("%s", movie.directores[enesimo]);
            getchar();
```

```
        }
        arreglo[iesimo] = movie;
    }
}

void imprimirArreglo(struct pelicula arreglo [TAM]){
    int iesimo, enesimo;
    printf("####### Contenido del arreglo #######\n");
    for (iesimo=TAM-1 ; iesimo>=0 ; iesimo--) {
        printf("####### Película %d #######\n", iesimo+1);
        printf("PELÍCULA: %s\n", arreglo[iesimo].nombre);
        printf("GÉNERO: %s\n", arreglo[iesimo].genero);
        printf("AÑO: %d\n", arreglo[iesimo].anio);
        printf("DIRECTOR(ES):\n");
        for (enesimo=0 ; enesimo<arreglo[iesimo].numDirectores ;
enesimo++){
            printf("%s\n", arreglo[iesimo].directores[enesimo]);
        }
    }
}
```

Como se puede observar en el ejemplo anterior, un tipo de dato abstracto permite modelar cualquier elemento (película en este caso) para ser manipulado como una unidad dentro de una estructura de datos. Los elementos (estructuras en lenguaje C) pueden estar compuestos por tipos de datos simples (enteros, caracteres o reales) o por datos compuestos (arreglos y otras estructuras en lenguaje C), haciendo que el modelado de elementos sea tan específico como se requiera.

Bibliografía



El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.