Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Antonio Ramos González

Matrícula: 372576

Maestro: Pedro Núñez Yépiz

Actividad No. 10: ACTIVIDAD 10

Tema - Unidad 7: Estructuras

Ensenada Baja California a 15 de octubre del 2023

1. INTRODUCCIÓN

Los tipos de datos structs son un mecanismo que nos permiten crear nuestros propios tipos de datos personalizados, permitiéndonos organizar distintos tipos de datos (como lo podrían ser cadenas, números enteros, números flotantes, etc) dentro de una sola variable.

El typedef es una "funcion" o "característica", que nos permite a nosotros como programadores crear nuestros propios tipos de "variables", permitiéndonos un mejor manejo de los datos durante la ejecución del programa.

2. COMPETENCIA

Lo que se busca con esta práctica es que el alumno siente las bases sobre las estructuras, las cuales son indispensables en la programación para la organización de distintos tipos de datos. También la creación de distintos tipos de datos, facilitando así el uso de los structs. También se buscó reforzar los distintos conocimientos obtenido en prácticas anteriores, usando distintos tipos de ciclos, funciones y estructuras de control, así como distintos tipos de validaciones

3. FUNDAMENTOS

Las estructuras en c son colecciones de variables que se referencian bajo un mismo nombre. Dichas estructuras proporcionan una manera conveniente para manejar información estrechamente relacionada. Normalmente, los datos dentro de un estruct están relacionados lógicamente con otros, como lo podrían ser los datos de un alumno. La menra de de clarar un estruct es:

Struct alumno{ primero se define el struct con una etiqueta llamada alumno

Char nombre[10];

Int matricula; dentro de las llaves irán los tipos de datos

Char apellidoP[15];

Char apellidoM[15];

}info_alumno; fuera del corchete se declara el nombre de la variable. En dicho lugar se pueden definir más de un tipo de variable

Typedef se puede usar oara definir nuevos nombres de datos de la siguiente manera:

Typedef <tipo de dato> <nombre>

Typedef se puede usar para crear nombres de tipos de datos más complejos, como lo sería el estruct

typedef Struct alumno{ primero se define el struct con una etiqueta llamada alumno

Char nombre[10];

Int matricula; dentro de las llaves irán los tipos de datos

Char apellidoP[15];

Char apellidoM[15];

}info_alumno;

Una vez declarado el nuevo tipo de dato ya podría ser usado como un tipo de dato normal, por ejemplo definiéndolo como vector

Info_alumno arreglo[500];

4. PROCEDIMIENTO

REALICE EL SIGUIENTE PROGRAMA QUE CONTENGA UN MENÚ.

MENÚ
1.- AGREGAR (AUTOM 10 REGISTROS)
2.- AGREGAR MANUAL
3- ELIMINAR REGISTRO (lógico)
4.- BUSCAR
5- ORDENAR
6.- IMPRIMIR
0.- SALIR

UTILIZAR UN ARREGLO DE 500 REGISTROS
SE DEBERÁ **UTILIZAR ESTRUCTURAS** CON LOS DATOS BÁSICOS DE UN ALUMNO (status, Matricula, ApPat, ApMat, Nombre, Edad, Sexo)
Busqueda y Ordenacion por campo MATRICULA

nota: usar librería propia

5. RESULTADOS Y CONCLUSIONES

Desde mi punto de vista tanto el uso del struct como el uso de tipedef son fundamentales para que los programadores puedan tener un mayor manejo de grandes cantidades de datos que estén estrechamente relacionados, pudiéndolos guardar dentro de solo un tipo de dato sin necesidad de crear más tipos de datos.

```c
// Antonio Ramos Gonzalez Mt: 372576
// 10/9/2023 || 10/13/2023
// En esta Practica se definira un tipo de variable como un struct, para ingresar datos de distintos alumnos con un menu
// RGA_Act10_932
#include "Babilonia.h" //llamar mi biblioteca
#define N 500
typedef struct _alumn
{
    char name[30];
    char lasP[30]; // apellido paterno
    char lasM[30]; // apellido materno
    int mt;        // matricula
    int age;       // edad
    char sex[10];  // sexo
    int status;    // dado de baja o no
} Talum;           // defino mi tipo de bariable de tipo struct

int msg();
void nombAl(char nombre[], int sex);
void apellidoAl(char apellido[]);
void menu();
void printReg(Talum alumn[], int n);
int order_reg(Talum alumn[], int n, int band);
int searchSec(Talum alumn[], int n, int mt);
int searchBin(Talum alumn[], int inf, int sup, int mt);
Talum genAl(Talum alumn[], int i);
Talum llenarManual(Talum alumn[], int i);
Talum eliminar(Talum alumn);
void busq(int ind);
int verMt(Talum alumn[], int n, int mt);

int main()···

// Muestra mensaje
int msg()···

// Meno de elecciones
void menu()···
```

```c
int main()···

// Muestra mensaje
int msg()···

// Meno de elecciones
void menu()···

// Funcion para nombres
void nombAl(char nombre[], int sex)···

// Funcio para apellidos
void apellidoAl(char apellido[])···

// Funcion que verifica que no se repitan matriculas
int verMt(Talum alumn[], int n, int mt)···

// Funcion que genera datos de alumnos aleatorios
Talum genAl(Talum alumn[], int i)···

// Funcion para ingresar a un alumno de manera manual
Talum llenarManual(Talum alumn[], int i)···

// Funcion de busqueda secuencial
int searchSec(Talum alumn[], int n, int mt)···

// Funcion que ordena el vector
int order_reg(Talum alumn[], int n, int band)···

// Funcion para imprimir el vector
void printReg(Talum alumn[], int n)···

// Funcion que imprime el alumno buscado
void busq(int ind)···

// Funcion de busqueda binaria
int searchBin(Talum alumn[], int inf, int sup, int mt)···
```

6. ANEXOS

Link GitHub:
https://github.com/Anrago/Programacion-estructurada/tree/main/Actividad_10

7. REFERENCIAS

Diseño de algoritmos y su codificación en lenguaje C

Corona, M.A. y Ancona, M.A. (2011)..

España: McGraw-Hill.

ISBN: 9786071505712

Programación estructurada a fondo:implementación de algoritmos en C

:Pearson Educación.Sznajdleder, P. A. (2017)..

Buenos Aires,Argentina: Alfaomega

Como programar en C/C++

H.M. Deitel/ P.J. Deitel

Segunda edición

Editorial: Prentice Hall.

ISBN:9688804711

Programación en C.Metodología, estructura de datos y objetos

Joyanes, L. y Zahonero, I. (2001)..

España:McGraw-Hill.

ISBN: 8448130138