UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Paradigmas De La Programacion

PRACTICA 1

Elementos básicos de los lenguajes de programación **Sistema de Biblioteca en C**.

INTRODUCCION

En esta practica identificaremos algunas partes de codigo el cual nos ayudara a mejorar su comprencion no solamente para este codigo si no oara todos los codigos en particular.

1. Nombres (Identificadores)

Son los que usamos para variables, funciones, estructuras, etc.

Ejemplos del código:

```
static int static_var;
int bss_var;
book_t *library;
member_t *members;
int bookCount, memberCount;
```

2. Marcos de activación (Activation Records / Stack Frames) Cada vez

que se llama una función, se crea un marco en el **stack** con sus variables locales y parámetros.

Ejemplo:

```
void addBook(book_t **library, int* count) {
 book_t *new_book = (book_t *)malloc(sizeof(book_t)); // variable local en el
stack
}
```

3. Bloques de alcance (Scope Blocks)

Los bloques `{ ... }` delimitan el **alcance** de variables y comandos.

Ejemplo en `main()` y en `switch`:

```
switch (choice) {
 case 1: {
 addBook(&library, &bookCount);
 break;
 }
}
```

4. Administración de memoria

Tu programa maneja **stack, heap y segmentos de datos**:

- **Stack**: variables locales (`choice`, `bookCount`, `memberCount`).
- **Heap**: asignación dinámica con `malloc`, `realloc`, `free`.

```
book_t *new_book = (book_t *)malloc(sizeof(book_t));
free(current);
```

- **Segmento de datos**:
  - Variable estática: `static int static_var`.
  - Variable global no inicializada (BSS): `int bss_var;`.

5. Expresiones

Son combinaciones de operadores, variables y funciones que generan un

valor. **Ejemplo:**

```
memberFound->issued_books = realloc(memberFound->issued_books, memberFound-
>issued_count * sizeof(int));
```

6. Comandos (Sentencias)

Son instrucciones ejecutables.

Ejemplo:

```
printf("El libro fue agregado exitosamente!\n");
```

```
(*count)++;
```



```
return;
```

7. Control de secuencia

a) Selección (if/switch)

```
if (!library) {
 printf("\nNo hay libros disponibles.\n"); }
```

```
switch (choice) {
 case 1: addBook(&library, &bookCount); break; }
```

b) Iteración (for, while, do...while)

```
while (current) {
 current = current->next;
}
```

```
for (int i = 0; i < current->issued_count; i++) {
printf("Libro ID: %d\n", current->issued_books[i]); }
```

c) Recursión

La función `displayBooksRecursive()` se llama a sí misma:

```
void displayBooksRecursive(book_t *library) {
if (!library) return;
 ...
 displayBooksRecursive(library->next);
}
```

8. Subprogramas (Funciones)

Son las funciones definidas para modularizar el

código. **Ejemplos:**


```
void addBook(book_t **library, int* count);
book_t* findBookById(book_t *library, int bookID);
void displayBooks(book_t *library);
```

9. Tipos de datos

El código usa varios tipos de datos:

Primitivos: `int`, `char`, `float` (indirectamente en `scanf/printf`).
Compuestos / Definidos por el usuario:

```
typedef struct _book { ... } book_t;
typedef struct _member { ... } member_t;
typedef enum { FICTION, NON_FICTION, ... } genre_t;
```

Punteros: `book_t *library`, `member_t *members`.

Conclucion

Es importante comprender bien lo que hace cada fragmento de codigo para asi en futuros codigos ya sea que tu los hagas o otros ya echos sepas como estan estructurados y lo comprendas de una mejor manera.

Upload files · israelalcala/Portafolio-de-Evidencias-