ALGORITMO DE SELECCIÓN y ALGORITMO DE INSERCION

Autores:

- Alfonso Alejandro Arroyo Paredes
- Hans Abel Benalcázar Cuastuza
- Luis Anibal Lema Toabanda
- Cristhofer Jose Noroña Ramírez
- Roberth Isaac Segovia Morales
- Josué Bryan Almeida Navarrete
- Johnny Kevin Becerra Gutierrez

Fecha de presentación: 2023/07/03
NRC: 8753
Tutor: Omar Rolando Quimbita Chiluisa
INVESTIGACIÓN BIBLIOGRÁFICA
TEMA: Librerías

RESUMEN:

Las librerías, bibliotecas son fundamentales al momento de crear un programa en C ya que son un componente esencial, puesto que sin ellas no se podrá dar un total desarrollo del programa al momento de ejecutarlo y compilarlo.

Como nos menciona el artículo sobre librerías de la página web abrirarchivos.info “una librería es un conjunto de funciones predefinidas que se pueden utilizar en un programa para realizar tareas específicas. Estas funciones son escritas por otros programadores y se pueden incluir en el código fuente de un programa para realizar operaciones avanzadas sin tener que escribir todo el código desde cero.” (Celesta, 2023).

Además, cuando queremos programar una librería en el lenguaje C, antes que todo escribimos nuestro código para nuestras funciones que deseemos escribir, para luego compilarlo en un archivo de objetivo de objeto y vincularlo con el programa principal, y ya podremos utilizar la librería en cualquier programa que lo necesitemos.

OBJETIVO GENERAL:

- Comprender a fondo el concepto y la importancia de las librerías en el lenguaje de programación C, así como su papel fundamental en el desarrollo, optimización y reutilización de código, permitiendo la ejecución eficiente de programas mediante la inclusión de funciones predefinidas que realizan tareas específicas y avanzadas.

OBJETIVOS ESPECÍFICOS:

- Adquirir un conocimiento profundo sobre la inclusión de librerías en el código fuente de un programa en C, reconociendo su capacidad para realizar operaciones avanzadas sin necesidad de escribir todo el código desde cero.

- Dominar el proceso de compilación y vinculación de una librería en el programa principal en C, comprendiendo cómo se generan archivos de objeto y cómo se integran con el programa principal para utilizar las funciones de la librería.

- Valorar la importancia estratégica de las librerías como herramientas fundamentales en la programación en C, reconociendo su capacidad para mejorar la eficiencia, la legibilidad y el mantenimiento del código, así como para acelerar el proceso de desarrollo de software.

INTRODUCCIÓN:

Una librería es un archivo que el compilador puede leer y donde encontrará una serie de instrucciones para realizar funciones, estas dependen directamente de la librería que se utilice, para no generar un código muy largo, es necesario identificar y aplicar únicamente las librerías a utilizar.

Las librerías más utilizadas en C son la librería estándar de C y la librería de entrada/salida de C. La librería estándar de C proporciona funciones para trabajar con caracteres, cadenas, memoria, entrada/salida, y otras tareas comunes en la programación en C. La librería de entrada/salida de C proporciona funciones para leer y escribir archivos, así como para interactuar con la entrada y la salida del usuario.

Las librerías en programación PDF son un conjunto de funciones que se utilizan para trabajar con archivos PDF. Estas librerías proporcionan funciones para leer, escribir y editar archivos PDF, así como para agregar imágenes, gráficos y otros elementos a los documentos PDF.

Entonces, las librerías son un conjunto de funciones predefinidas que se utilizan en un programa para realizar tareas específicas. En C, las librerías se utilizan para interactuar con los sistemas operativos y realizar tareas avanzadas sin tener que escribir todo el código desde cero. En C + +, las librerías también incluyen clases y plantillas. Las librerías más utilizadas en C son la librería estándar de C y la librería de entrada/salida de C. Las librerías en programación PDF se utilizan para trabajar con archivos PDF.

MARCO TEÓRICO:

Las librerías son un conjunto de funciones predefinidas que se utilizan en un programa para realizar tareas específicas. En C, las librerías se utilizan para interactuar con los sistemas operativos y realizar tareas avanzadas sin tener que escribir todo el código desde cero. En C + +, las librerías también incluyen clases y plantillas. Las librerías más utilizadas en C son la librería estándar de C y la librería de entrada/salida de C.

La librería de C para la gestión de tiempos

Uno de los problemas de gran complejidad que tenemos en todos los lenguajes es la gestión del tiempo, por ello para C tenemos la librería time.h

La librería para las operaciones matemáticas

Aunque C posee algunas operaciones matemáticas básicas definidas en el propio lenguaje, como la resta o la multiplicación, no contiene operaciones matemáticas avanzadas. Para ello vamos a utilizar la librería math.h

La librería para la gestión de cadenas de caracteres en C

Las cadenas de caracteres, por su forma de constituirse en C, siempre dan más problemas que los tipos numéricos. Por ello es muy recomendable usar la librería string.h para manejarlas correctamente.

Biblioteca estándar

Las funciones, tipos y macros de la biblioteca estándar están declarados en headers estándar:

<assert.h> <float.h> <math.h> <stdarg.h> <stdlib.h>

<ctype.h> <lim its.h> <setjmp.h> <stddef.h> <string.h>

<errno.h> <locale.h> <signal.h> <stdio.h> <time.h>

Se puede tener acceso a un header por medio de

#include <header>:

Los headers se pueden incluir en cualquier orden y número de veces, un header se debe incluir fuera de cualquier declaración o definición externa y antes de

cualquier uso de cualquier cosa que declare. No es necesario que un header sea un archivo fuente.

Los identificadores externos que principian con subguión están reservados para uso de la biblioteca, como lo están todos los otros identificadores que principian con un subguión y una letra mayúscula u otro subguión

Entrada y salida: <stdio.h>:

Las funciones, tipos y macros de entrada y salida, definidos en < stdio.h>, representan cerca de la tercera parte de la biblioteca.

Funciones para cadenas: < string .h >:

Hay dos grupos de funciones para cadenas definidas en el header < string.h>. Las

Las primeras tienen nombres que comienzan con str; las segundas tienen nombres que comienzan con mem. Excepto para memmove, el comportamiento está indefinido si la copia ocurre entre objetos que se traslapan.

Funciones matemáticas: < math .h >:

La función math.h en C es una biblioteca estándar que proporciona diversas funciones matemáticas para realizar cálculos numéricos. Estas funciones incluyen operaciones algebraicas, trigonométricas, exponenciales y de redondeo, entre otras. Permiten realizar cálculos complejos y precisos en programas de C. Además, math.h también incluye constantes matemáticas como π y e. En resumen, esta biblioteca es utilizada para realizar operaciones matemáticas avanzadas en programas de C, facilitando el desarrollo de algoritmos y la manipulación de datos numéricos con precisión y eficiencia.

Funciones de utilería: < stdlib .h >:

El header <stdlib.h> declara funciones para conversión numérica, asignación de memoria y tareas semejantes.

double a to f (const char *s)

atof convierte s a double; es equivalente astrid (s , (ch a r* *)NULL).

Listas de argumentos variables: < stdarg .h >:

El header <stdarg.h> proporciona recursos para recorrer una lista de argumentos de función de tamaño y tipo desconocido.

Supóngase que las targ es el último parámetro nombrado de una función f con un número variable de argumentos. Entonces se declara dentro de f una variable ap de tipo va_list que apuntará a cada argumento en orden:

Saltos no locales: < setjmp .h >:

Las declaraciones que están en <setjmp.h> proporcionan una forma de evitar la secuencia normal de llamada y regreso de funciones, típicamente para permitir un regreso inmediato de una llamada a una función profundamente anidada.

Señales: < signal.h >:

Hl header < signal.h > de facilidades para manejar condiciones excepcionales que aparecen durante la ejecución, tal como una señal de interrupción de una fuente externa o un error en la ejecución.

Funciones de fecha y hora < time .h >

El header < time.h > declara los tipos y funciones para manipulación de fecha y hora. Algunas funciones procesan la hora local, que puede diferir de la del calendario, por ejemplo, debido a la zona horaria, clock.t y time.t son tipos aritméticos que representan tiempos, y struct tm mantiene las componentes de un calendario:

Límites definidos en la implantación: <limits.h > y < float.h >

El header < limits.h > define constantes para el tamaño de los tipos enteros. Los valores mostrados son magnitudes mínimas aceptables; se pueden emplear valores mayores

DESARROLLO:

Librería <stdlib.h>

La librería **<stdlib.h>** es un componente fundamental en el mundo de la programación. Dentro de esta librería se encuentran diversas funciones estándar que nos permiten realizar tareas comunes y necesarias en el desarrollo de software.

Además, una amplia gama de funciones que facilitan el desarrollo de aplicaciones. Estas funciones proporcionan herramientas poderosas para abordar problemas de programación de manera más eficiente y estructurada. Al comprender y utilizar correctamente estas funciones, los programadores pueden optimizar el rendimiento de sus programas y mejorar la productividad en el desarrollo de software.

Estas funciones abarcan desde la conversión de cadenas y generación de números aleatorios hasta la gestión de memoria dinámica y el ordenamiento de elementos. De la siguiente manera: (Author, 2020) “Las funciones que pertenecen a stdlib. h pueden clasificarse en las siguientes categorías: conversión, memoria, control de procesos, ordenación y búsqueda, matemáticas.”

Comprender estas funciones nos brindará herramientas poderosas para abordar problemas de programación de manera más eficiente y estructurada.

Las funciones de conversión de cadenas (atoi, atof, atole) nos permiten convertir datos almacenados en formato de texto a números enteros, de punto flotante o de tipo long, respectivamente. Estas funciones son útiles cuando necesitamos extraer valores numéricos de cadenas, como cuando leemos datos de un archivo o recibimos entradas del usuario.

Ejemplo con la función atoi:

```
#include <stdlib.h>
#include <stdio.h>
int main() {
    char numStr[] = "123";
    int num = atoi(numStr);
```

```
    printf("El número convertido es: %d\n", num);
    return 0;
}
```

Ejemplo con la función atof:

```
#include <stdlib.h>
#include <stdio.h>

int main() {
    char floatStr[] = "3.14";
    float num = atof(floatStr);
    printf("El número convertido es: %.2f\n", num);
    return 0;
}
```

La función rand nos proporciona una manera sencilla de generar números aleatorios en un rango específico. Al llamar a rand, se genera un número pseudoaleatorio. Podemos controlar la secuencia de números generados utilizando la **función srand**, que establece la semilla para la generación de números aleatorios. Esto nos permite obtener secuencias de números repetibles o aleatorias según nuestra necesidad.

Ejemplo con la función rand y srand:

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
int main() {
    srand(time(NULL));  // Establecer la semilla basada en el tiempo actual
    int num = rand() % 10;  // Generar un número aleatorio entre 0 y 9
    printf("El número aleatorio es: %d\n", num);
    return 0;
```

```
}
```

En el tema de **asignación de memoria dinámica**, las funciones malloc, calloc y realloc son fundamentales. La función malloc nos permite asignar un bloque de memoria de tamaño específico durante la ejecución del programa. La función calloc es similar a malloc, pero también inicializa el bloque de memoria asignado con ceros. La función realloc se utiliza para modificar el tamaño de un bloque de memoria previamente asignado. Estas funciones nos brindan la flexibilidad necesaria para manejar estructuras de datos dinámicas y optimizar el uso de recursos en nuestros programas.

Ejemplo con las funciones de asignación de memoria dinámica:

```
#include <stdlib.h>
#include <stdio.h>
int main() {
    int *ptr = (int *)malloc(sizeof(int));  // Asignar memoria para un entero
    *ptr = 10;  // Asignar valor al entero
    printf("El entero asignado es: %d\n", *ptr);
    free(ptr);  // Liberar la memoria asignada
    return 0;
}
```

Las funciones de ordenamiento y búsqueda, como qsort y bsearch, nos ayudan a ordenar y buscar elementos en arreglos de manera eficiente. La función qsort utiliza el algoritmo de ordenamiento rápido (quick sort) para ordenar los elementos de un arreglo en un orden específico. Por otro lado, la función bsearch realiza una búsqueda binaria en un arreglo ordenado para encontrar un elemento en particular. Estas funciones son esenciales para organizar y acceder a datos de manera eficiente en diversas aplicaciones.

Ejemplo de las funciones de ordenamiento y búsqueda:

```
#include <stdlib.h>
#include <stdio.h>
```

```
// Función de comparación para qsort
int comparar(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
int main() {
    int arreglo[] = {5, 2, 8, 1, 9};
    int n = sizeof(arreglo) / sizeof(arreglo[0]);

    qsort(arreglo, n, sizeof(int), comparar);

    printf("Arreglo ordenado: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arreglo[i]);
    }
    printf("\n");

    return 0;
}
```

Otras **funciones útiles** presentes en <stdlib.h> son **exit, system y getenv.** La función exit se utiliza para finalizar la ejecución de un programa de manera controlada y devolver un valor al sistema operativo. La función system nos permite ejecutar comandos del sistema operativo desde un programa. Por último, la función getenv se utiliza para obtener el valor de una variable de entorno del sistema operativo, lo que brinda acceso a información relevante o configuraciones específicas del entorno. Estas funciones amplían nuestras posibilidades y nos permiten interactuar con el entorno que rodea a nuestra aplicación.

Ejemplo de otras funciones;

- ***Ejemplo con la función exit:***

```
#include <stdlib.h>
#include <stdio.h>
int main() {
    int resultado = 42;
    printf("Finalizando programa...\n");
    exit(resultado);
}
```

- ***Ejemplo con la función system:***

```
#include <stdlib.h>
#include <stdio.h>
int main() {
    printf("Ejecutando comando del sistema...\n");
    system("ls -l");  // Ejecutar comando "ls -l" en sistemas Unix
    return 0;
}
```

- ***Ejemplo con la función getenv:***

```
#include <stdlib.h>
#include <stdio.h>

int main() {
    char *valor = getenv("HOME");  // Obtener valor de la variable de entorno "HOME"
    printf("El valor de la variable HOME es: %s\n", valor);
    return 0;
```

```
}
```

RESULTADOS:

Para el ejemplo se utilizó como referencia una lotería nacional, por lo cual necesitábamos, el nombre del propietario del boleto y los 3 números aleatorios en un boleto de lotería y que antes de generarse en pantalla el nombre y los números, se pause el programa por lo cual el enunciado quedó así:

Realizar un código que pida el nombre al usuario y que genere tres números (aleatorio)para un boleto de lotería para dicho usuario, este debe de tener una pausa entre el ingreso del nombre y la generación en pantalla.

```
#include<stdio.h>
#include<stdlib.h>

/*
Realizar un codigo que pida el nombre al usuario y que genere tres numeros
(aleatorio)para un boleto de loteria para dicho usuario, este debe de tener
una pausa entre el ingreso del nombre y la generación en pantalla.
*/

int main (){
        int n,n2,n3;
        n3=1+rand()%(100-1+1);
        n2=1+rand()%(100-1+1);
        n=1+rand()%(100-1+1);
        char Nom[10];
printf("Ingrese su nombre\n");
scanf("%s",&Nom);
```

```
system("pause");

printf("Su nombre es %s\n",Nom);

printf("y sus numeros de loteria son: %i,%i,%i ",n,n2,n3);


        return 0;

}
```

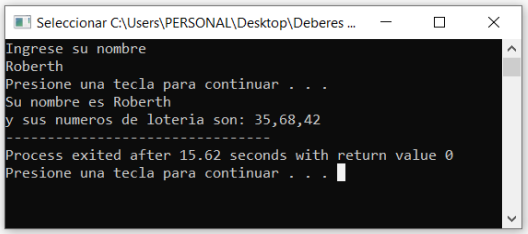
SistemPause.c

```c
#include<stdio.h>
#include<stdlib.h>

/*
Realizar un codigo que pida el nombre al ususario y que genere tres numeros
(aleatorio)para un boleto de loteria para dicho usuario, este debe de tener
una pausa entre el ingreso del nombre y la generación en pantalla.

*/
int main (){
    int n,n2,n3;
    n3=1+rand()%(100-1+1);
    n2=1+rand()%(100-1+1);
    n=1+rand()%(100-1+1);
    char Nom[10];
printf("Ingrese su nombre\n");
scanf("%s",&Nom);
system("pause");
printf("Su nombre es %s\n",Nom);
printf("y sus numeros de loteria son: %i,%i,%i ",n,n2,n3);

    return 0;
}
```



DISCUSIONES:

El código muestra la aplicación de 2 funciones de la librería stdlib.h (librería especializada en: Gestión de memoria dinámica, Control en los procesos y funciones matemáticas), las funciones mencionadas son:

rand: dicha función nos sirve para generar un número aleatorio, con parámetros tal como se ve

i+rand()%(100-i+1);

i: el límite inferior del número randómico.

%(100): el límite máximo del número randómico.

y como comentario en (100-i+1) el +1 nos sirve para poder generar en pantalla inclusive el límite superior.

system("pause");

Dicha función nos genera un pausa en el programa, dicho programa no se reanudará mientras no se teclee cualquier tecla en el computador.

Este programa ha sido diseñado y desarrollado de manera exhaustiva para asegurar que cumpla con todos los requerimientos solicitados en el enunciado del proyecto. Se ha prestado especial atención a la simplicidad y claridad del código, garantizando así que sea fácil de entender para cualquier persona que lo revise. Además, se ha tenido en cuenta la optimización del programa, asegurándose de que su tamaño sea lo más reducido posible sin comprometer su funcionalidad. Esto facilita su mantenimiento y posibles actualizaciones en el futuro. En resumen, este programa es una solución eficiente y efectiva que cumple con los requerimientos establecidos, al tiempo que proporciona una estructura compacta y comprensible.

CONCLUSIONES:

En este informe vimos la funcionalidad y utilidad que conlleva esta librería para el desarrollo de códigos, proyectos en C. determina si la librería cumple con el propósito planteado en este tema, así ofrecer características necesarias para abordar problemas ya anteriormente planteados. Además, hemos averiguado que tenemos la capacidad suficiente de crear nuestra propia librería, nos brinda la capacidad de personalizar y adaptar funciones específicas a nuestras necesidades. Esto nos puede beneficiar tanto académicamente como laboralmente reemplazando librerías ya existentes.

REFERENCIAS BIBLIOGRÁFICAS:

1. KERNIGHAN DENNIS, B. W. M. R. (1991). EL LENGUAJE DE PROGRAMACIÓN: Vol. BRIAN W. KERNIGHAN DENNIS M. RITCHIE (SEGUNDA EDICIÓN). UNIX es una marca registrada AT & T. https://frrq.cvg.utn.edu.ar/pluginfile.php/13741/mod_resource/content/0/El-lenguaje-de-programacion-C-2-ed-kernighan-amp-ritchie.pdf

2. Durán, J. (2017, 3 diciembre). ▷ Las librerías más usadas en C. Lenguaje de programación. https://lenguajedeprogramacion.com/programacion-c/librerias-mas-usadas-c/#La_libreria_de_C_para_la_gestion_de_tiempos

3. colaboradores de Wikipedia. (2023). Biblioteca estándar de C. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Biblioteca_est%C3%A1ndar_de_C

4. Rivel_co. (2020, 21 octubre). Librerías para C++. Include Poetry. https://www.include-poetry.com/Code/C++/Introduccion/Librerias

5. Gómez, P. (2021, 27 octubre). Qué es una librería en programación - DevCamp. DevCamp. https://devcamp.es/que-es-libreria-programacion/

6. Rivel_co. (2020b, octubre 21). Librerías para C++. Include Poetry. https://www.include-poetry.com/Code/C++/Introduccion/Librerias/