	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 20

No. de práctica(s): 06

Integrante(s): Juan Manuel Cuellar Orbezo

No. de lista o brigada: 07

Semestre: 2024 - 2

Fecha de entrega: 19 Marzo 2024

Observaciones:

CALIFICACIÓN: _____

Práctica 06 - Entorno y fundamentos del lenguaje C

Objetivo:

El alumno elaborará programas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

Desarrollo

Editor Visual Interface de GNU/Linux (vi)

Aprender a programar en C usando un editor VI, en lugar de un compilador integrado es desafiante. Aunque requiere un mayor esfuerzo inicial, ofrece beneficios a largo plazo.

VI permite editar código con precisión. Su ligereza y rapidez lo hacen ideal para trabajar en máquinas con recursos limitados. Asimismo dominar VI brinda una valiosa habilidad para editar cualquier tipo de archivo de texto en cualquier plataforma. Además, ayuda a comprender mejor el proceso de compilación al escribir manualmente los comandos.

Sin embargo, hay que considerar que la curva de aprendizaje puede ser intimidante al principio, requiriendo tiempo y práctica para dominar sus comandos. VI carece de características presentes en IDEs modernos, como resaltado de sintaxis, autocompletado y depuración. Además de una mayor posibilidad de errores; escribir código en VI sin asistencia aumenta la probabilidad de cometer errores de sintaxis.

Además de los comandos mencionados en prácticas anteriores se deben mencionar estos otros tres, que son importantes para ejecutar programas en lenguaje C escritos en VI.

- **vi nombre_archivo[.ext]** - Inicia el editor de texto VI desde una ventana de terminal
- **gcc ejemplo.c** - compila el código fuente del programa C "ejemplo.c" a un ejecutable llamado "a.out" (o "a.exe" en Windows).
- **./ejemplo.out** - Ejecuta el programa creado por el usuario

Ejercicio 1 - Tipos de variables

Este programa muestra en pantalla los valores de variables de diferentes tipos de datos: entero, flotante, doble y caracter. Además, demuestra el uso de diferentes formatos de salida para cada tipo de dato. Así como la función printf para dar formato a la salida y controlar la presentación de los datos.

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

```c
int main() {

    //Declaración de variables
    int entero;
    float flotante;
    double doble;
    char caracter;

    //Asignación de variables
    entero = 14;
    flotante = 3.5f;
    doble = 6.8e10;
    caracter = 'A';

    //Funciones de salida de datos en pantalla
    printf("La variable entera tiene valor: %i \n", entero);
    printf("La variable flotante tiene valor: %f \n", flotante);
    printf("La variable doble tiene valor: %f \n", doble);
    printf("La variable caracter tiene valor: %c \n", caracter);
    printf("Entero como octal: %o \nComo Hexadecimal %X \n", entero, entero);
    printf("Flotante con precisión: %5.2f \n", flotante);
    printf("Doble con precisión: %5.2f \n", doble);
    printf("Carácter como entero: %d \n", caracter);
    return 0;
}
```

Ejercicio 2 - Entrada de datos

El programa solicita al usuario que ingrese un valor entero y un valor real. Luego, imprime el valor de las variables con diferentes formatos.

Además, el programa convierte un valor entero a un caracter usando la tabla ASCII. El caracter A tiene un valor ASCII de 65, que se puede convertir a un caracter usando la operación de casting.

```c
#include <stdio.h>
int main()
{
int enteroNumero;
char caracterA = 65; // Convierte el entero a carácter ASCII.
double puntoFlotanteNumero;
// Asignar valor de teclado a una variable.
printf("Escriba un valor entero: ");
scanf("%i", &enteroNumero);
printf("Escriba un valor real: ");
```

```
scanf("%lf", &puntoFlotanteNumero);
// Imprimir valores con formato.
printf("\nImprimiendo las variables enteras \a\n");
printf("\t Valor de enteroNumero = %i \a\n", enteroNumero);
printf("\t Valor de caracterA = %c \a\n", caracterA);
printf("\t Valor de puntoFlotanteNumero = %lf \a\n",
 puntoFlotanteNumero);
printf("\t Valor de enteroNumero en base 16 = %x \a\n", enteroNumero);
printf("\t Valor de caracterA (65) en código hexadecimal = %x\n", caracterA);
printf("\t Valor de puntoFlotanteNumero\n");
printf("en notación científica = %e\n", puntoFlotanteNumero);
return 0;
}
```

Ejercicio 3 - Entrada de datos

El programa realiza operaciones aritméticas y lógicas con variables de tipo "short" (enteros cortos) y muestra el resultado en pantalla. El programa también muestra mensajes en pantalla para identificar las operaciones que se están realizando.

En resumen, el programa es un ejemplo de cómo realizar operaciones aritméticas y lógicas con variables de tipo "short" en C.

```
#include <stdio.h>
int main()
{
 short ocho, cinco, cuatro, tres, dos, uno;

 // 8 en binario: 0000 0000 0000 1000
 ocho = 8;
 // 5 en binario: 0000 0000 0000 0101
 cinco = 5;
 // 4 en binario: 0000 0000 0000 0100
 cuatro = 4;
 // 3 en binario: 0000 0000 0000 0011
 tres = 3;
 // 2 en binario: 0000 0000 0000 0010
 dos = 2;
 // 1 en binario: 0000 0000 0000 0001
 uno = 1;
 printf("Operadores aritméticos\n");
 printf("5 modulo 2 = %d\n",cinco%dos);
 printf("Operadores lógicos\n");
 printf("8 >> 2 = %d\n",ocho>>dos);
```

```
 printf("8 << 1 = %d\n",ocho<<1);
 printf("5 & 4 = %d\n",cinco&cuatro);
 printf("3 | 2 = %d\n",tres|dos);

 printf("\n");
 return 0;
}
```

Tarea

ASCII - Números

Todas las letras que se presentan en un editor de texto tienen por equivalente un número en ASCII. Por ejemplo el caracter “A” tiene un valor de 65. Si se imprime esta variable con:

- **%d e %i**: veremos el valor del carácter como un **entero decimal**. %d: 65.00 %i: 65
- **%o**: Imprimen el valor del carácter como un **entero octal**. %o: 101
- **%x**: Imprimen el valor del carácter como un **entero hexadecimal**. %x: 41

Características para crear una variable en C

La primera característica es el tipo de dato. Indicar qué tipo de información se almacenará en la variable. Enteros: int, short, long, etc. Reales: float, double, long double, etc. Caracteres: char Vacíos: void (para punteros genéricos)

El segundo aspecto es el nombre de la variable; Debe ser un nombre único que no se repita con otras variables, funciones o palabras clave del lenguaje. Tiene que comenzar con una letra y no debe usar espacios ni símbolos especiales. Y siempre se recomienda usar nombres descriptivos que representen el contenido de la variable.

Editor de Texto vs. Procesador de Texto

Característica	Editor de Texto	Procesador de Texto
Función principal	Editar texto plano sin formato.	Crear, editar y formatear documentos.
Formato	Sin formato, solo texto.	Soporte para diferentes formatos de fuente, color, tamaño, alineación, etc.

Funciones adicionales	Resaltado de sintaxis, autocompletado (en algunos editores).	Insertar imágenes, tablas, gráficos, encabezados, pies de página, etc.
Ejemplos	Bloc de notas (Windows), Notepad++ (Windows), Sublime Text (multiplataforma).	Microsoft Word (Windows/Mac), LibreOffice Writer (multiplataforma), Google Docs (online).
Uso ideal	Editar código fuente, scripts, páginas web.	Escribir documentos, cartas, informes, presentaciones.

Pseudocódigo que muestre la numeración del 1 al 1000

Variables de tipo char

El valor por defecto de un dato char en C depende de si es signed o unsigned:

1. char sin signo (unsigned char):

- El valor por defecto es 0.
- Puede almacenar valores entre 0 y 255 (2^8 - 1).
- Se utiliza para representar datos binarios, como códigos ASCII.

2. char con signo (signed char):

- El valor por defecto es indeterminado.
- Puede almacenar valores entre -128 y 127 (-2^7 a 2^7 - 1).
- Se utiliza para representar caracteres alfanuméricos y símbolos.

Corrimientos
El programa te permite calcular la potencia de 2 para un valor entero que introduces.

En lugar de usar una función de potencia o multiplicaciones repetitivas, utiliza el operador de desplazamiento a la izquierda (<<).

Este programa permite calcular la potencia de 2 para un valor entero introducido. En lugar de usar una función de potencia o multiplicaciones repetitivas, utiliza el operador de desplazamiento a la izquierda (<<). Esto para aprovechar la capacidad del computador de realizar operaciones bit a bit y calcular potencias de 2 de manera eficiente. Es importante recordar que este método solo funciona ya que usa el sistema binario; por lo cual buscar potencias con otras bases sería más complicado

```
#include <stdio.h>
int main()
{
 short uno;
 int x;
 int n;

 // 1 en binario: 0000 0000 0000 0001
 uno = 1;
 n = 0;

 printf("Hola! Seleccina un valor para obtener\nla potencia de 2 a ese valor\n");

 scanf("%i",& n);

 x = uno<<n;
```

```
 printf("2 ^ %d = ",n);
 printf("%d",x);
 return 0;
}
```

GitHub

A continuación, la liga al repositorio de GitHub - https://github.com/CuellarJM/practica6_fdp/

Conclusión

Este primer acercamiento a programar con C y poder usar estructuras secuenciales ha sido útil ya que se pone a prueba todo lo que se ha visto en el aula. Todos los pasos que hemos estado repasando durante este mes de clases han sido aplicados para este tipo de ejercicios. Es importante recalcar que el aprender a programar en C no es algo que se logre de la noche a la mañana y que puede tener una curva de aprendizaje considerable. Por lo cual este tipo de ejercicios y actividades son realmente valiosos.

Bibliografía

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

GitHub, Inc. (2024). Hello world. GitHub Docs. https://docs.github.com/en/get-started/start-your-journey/hello-world

Gookin, D. (2004). *C for dummies ; 2nd edition*. John Wiley & Sons.

Solano, J. A., García, E. E., & Montaño, L. S. (2022, February 21). Manual de prácticas del laboratorio de Fundamentos de programación. CDMX; Facultad de Ingeniería.