



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): No. 6 - Entorno y fundamentos del lenguaje C.

Integrante(s): López Olmedo Ernesto Yael

No. de lista o brigada: 27

Semestre: 2024-1

Fecha de entrega: 04 de Octubre 2023

Observaciones:

CALIFICACIÓN: _____

Introducción

Como continuación al laboratorio de fundamentos de programación ahora, el alumno se encuentra en una nueva fase del proceso de creación de un programa, siendo esta presentación en la que se empezará a ejecutar código en lenguaje C, con el cual se aplicarán las condiciones ya vistas en pseudocódigo, como la sintaxis específica, uso de sangría, uso de puntuación, entre otras más. Entonces, se realizarán ejemplos introductorios de funciones sencillas, declaración de variables, la impresión de pantalla y el escaneo de datos, partes fundamentales del desarrollo de programas. Con la finalidad que el alumno empiece a familiarizarse con el entorno de programación, su estructuración y al lenguaje en sí.

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

Entorno C:

El lenguaje programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un gran conjunto de operadores. No es un lenguaje de alto nivel, haciéndolo uno de los lenguajes más pequeños y sencillos de aprender, siendo un lenguaje potente con un campo de uso ilimitados.

Este lenguaje está estrechamente ligado con el sistema operativo UNIX, dado que fueron desarrollados conjuntamente, sin embargo, este lenguaje no opera ningún sistema operativo, ni dispositivo en concreto. Por esto se le llega a llamar como lenguaje de programación, por su utilidad para la creación/redacción de compiladores, sistema operativo o cualquier tipo de aplicación.

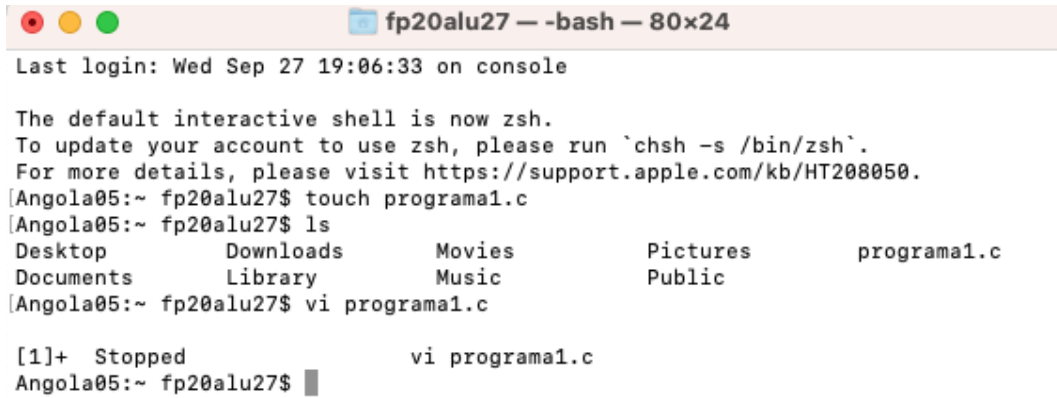
Palabras reservadas del lenguaje C:

1. *Auto* : Modificador que indica que una variable local se crea al inicio de la ejecución de la función y se destruye al final.
2. *Break* : Provoca que se termine la ejecución de la iteración o para salir de la sentencia switch, while o do/while, for e if.
3. *Switch Case*: Esta sentencia es una generalización de las sentencias if_else . Este operador es bastante similar a la anidación de varios if, el cual la expresión se evalúa como condición booleana, lo que dice es que solo hay dos valores posibles; en el caso del Switch, contiene múltiples soluciones y contiene un lector que facilita la selección.
4. *Char* : Tipo de dato o carácter alfanumérico (del tipo byte).
5. *Const* : Define variables cuyo valor debe permanecer constante durante toda la ejecución del programa, si se intenta cambiar su valor, dará lugar a un error interno.
6. *Int* : Tipo de dato entero.
7. *Continue* : Provoca que se comience una nueva iteración, evaluando la expresión de control.
8. *Default* : Es el caso por defecto que se ejecuta si dentro del switch no concuerda ninguno de los casos definidos.
9. *If...Else* : Permite la ejecución condicional de una sentencia. Si la condición es verdadera, se ejecutará la sentencia 1; si es falsa, se ejecutará la sentencia 2 (la cláusula else es opcional)

10. *While* : Sentencia de control iterativa, que evalúa una condición para su control. La sentencia es ejecutada repetitivamente mientras la condición sea verdadera.
11. *Do...while* : Variación de *while* donde se ejecuta y después se procede a evaluar la expresión de control. Repitiendo mientras la condición resulta verdadera; si no se define la condición, se asume como verdadera, provocando que se ejecute indefinidamente.
12. *For* : Sentencia de control iterativa que permite realizar saltos en el flujo de control de un programa, es decir, permite transferir el control del programa, alterando el flujo de control del mismo.
13. *Enum* : Permite declarar valores de datos que se ajustan a series ordenadas en las cuales un elemento sigue, o precede a otro *enum*.
14. *Extern* : Define que existe una variable global que está definida en otro archivo fuente.
15. *Float* : Las variables de este tipo almacenan números en formato de cómo flotante, esto es, contienen un valor de mantisa y otro de exponentes, codificando números con decimales.
16. *Long* : Tipo de dato entero largo con signo (normalmente 4 Bytes), para un valor entero largo.
17. *Return* : Esta es la sentencia de salida de una función, cuando se ejecuta se devuelve el control a la rutina que llamó a la función. Además, se usa para especificar el valor de retorno de la función.
18. *Static* : Modificador que indica que una variable local no se destruye al finalizar la función donde fue declarada.
19. *Signed* : Se usan para usar valores negativos y positivos o solo negativos.
20. *Void* : Es un tipo de dato que puede representar: nada(para funciones) o cualquier tipo de dato (para punteros).

- Laboratorio -

-Crear un programa :



```
fp20alu27 — -bash — 80x24
Last login: Wed Sep 27 19:06:33 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[Angola05:~ fp20alu27$ touch programa1.c
[Angola05:~ fp20alu27$ ls
Desktop          Downloads        Movies           Pictures         programa1.c
Documents        Library         Music           Public
[Angola05:~ fp20alu27$ vi programa1.c

[1]+  Stopped                  vi programa1.c
Angola05:~ fp20alu27$
```

Figura 1.1 - Creación de programa.c

-Abrir el documento:



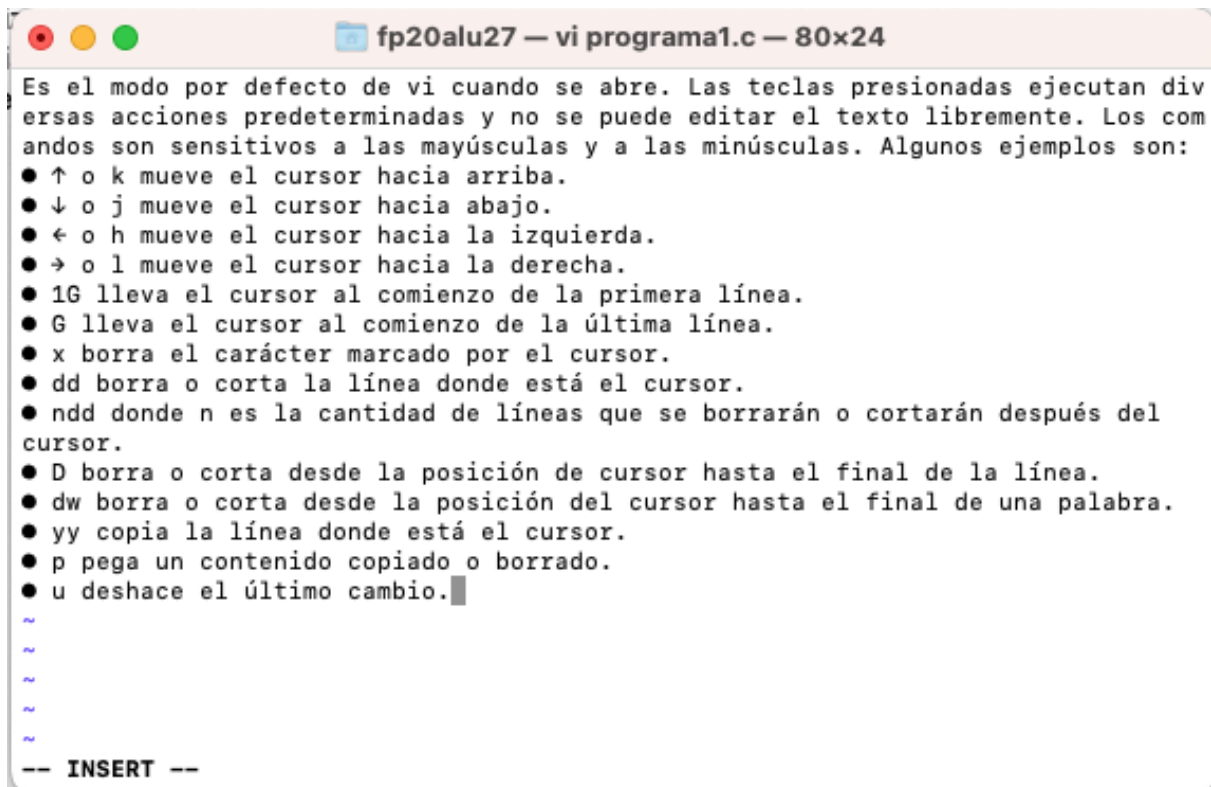
```
fp20alu27 — vi programa1.c — 80x24
-- INSERT --
```

Figura 1.2 - Apertura del programa.c

-Copiar y moverse dentro del programa:

A través del comando vi, permite ejecutar diversas acciones de movimiento antes de editarlo, los cuales son los siguientes:

- `↑` o `k` mueve el cursor hacia arriba.
- `↓` o `j` mueve el cursor hacia abajo.
- `←` o `h` mueve el cursor hacia la izquierda.
- `→` o `l` mueve el cursor hacia la derecha.
- `1G` lleva el cursor al comienzo de la primera línea.
- `G` lleva el cursor al comienzo de la última línea.
- `x` borra el carácter marcado por el cursor.
- `dd` borra o corta la línea donde está el cursor.
- `ndd` donde `n` es la cantidad de líneas que se borrarán o cortarán después del cursor.
- `D` borra o corta desde la posición de cursor hasta el final de la línea.
- `dw` borra o corta desde la posición del cursor hasta el final de una palabra.
- `yy` copia la línea donde está el cursor.
- `p` pega un contenido copiado o borrado.
- `u` deshace el último cambio.

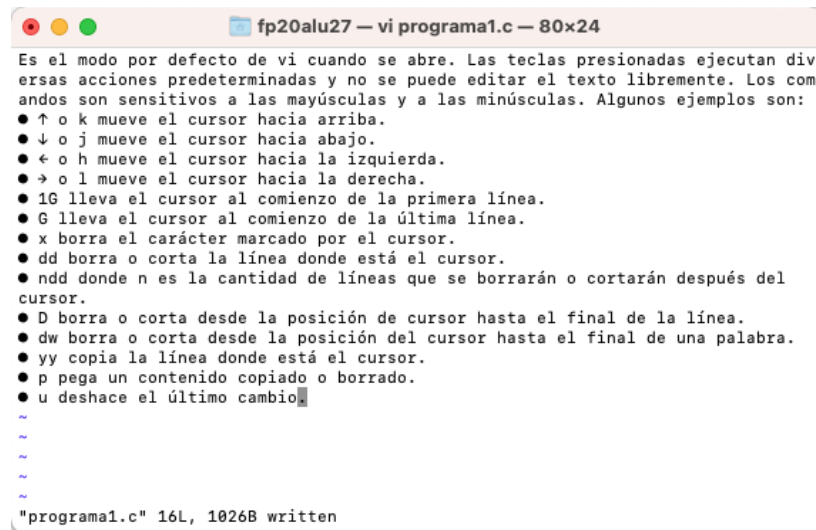


```
fp20alu27 — vi programa1.c — 80x24
Es el modo por defecto de vi cuando se abre. Las teclas presionadas ejecutan div
ersas acciones predeterminadas y no se puede editar el texto libremente. Los com
andos son sensitivos a las mayúsculas y a las minúsculas. Algunos ejemplos son:
● ↑ o k mueve el cursor hacia arriba.
● ↓ o j mueve el cursor hacia abajo.
● ← o h mueve el cursor hacia la izquierda.
● → o l mueve el cursor hacia la derecha.
● 1G lleva el cursor al comienzo de la primera línea.
● G lleva el cursor al comienzo de la última línea.
● x borra el carácter marcado por el cursor.
● dd borra o corta la línea donde está el cursor.
● ndd donde n es la cantidad de líneas que se borrarán o cortarán después del
cursor.
● D borra o corta desde la posición de cursor hasta el final de la línea.
● dw borra o corta desde la posición del cursor hasta el final de una palabra.
● yy copia la línea donde está el cursor.
● p pega un contenido copiado o borrado.
● u deshace el último cambio.
~
~
~
~
-- INSERT --
```

Figura 1.3 - Movimiento interno del archivo.

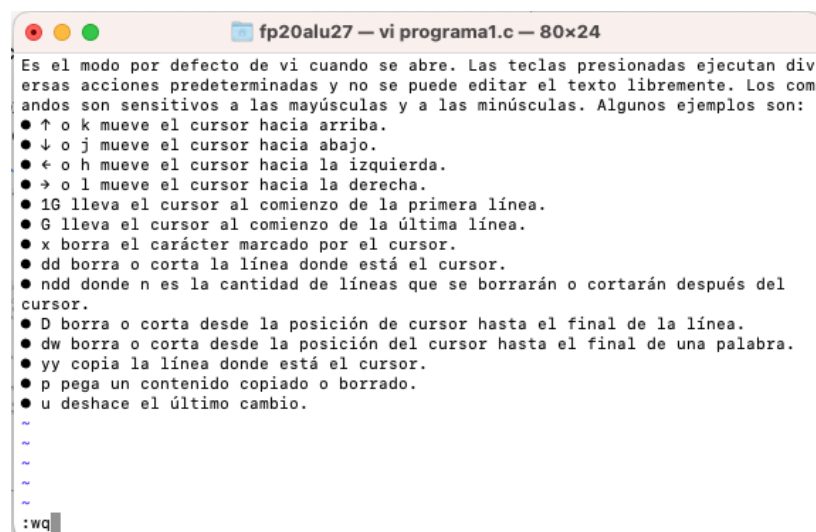
La siguiente lista de comandos son utilizados, terminada la edición del archivo, con ellos permiten entrar y salir del algoritmo, guardar o no los datos dentro de este. Nota: es posible cancelar un comando con la tecla escape (Esc).

- **/texto** donde la cadena **texto** será buscada hacia delante de donde se encuentra el cursor.
- **?texto** donde la cadena **texto** será buscada hacia atrás de donde se encuentra el cursor.
- **:q** para salir de *vi* sin haber editado el texto desde la última vez que se guardó.
- **:q!** para salir de *vi* sin guardar los cambios.
- **:w** para guardar los cambios sin salir de *vi*.
- **:w archivo** para realizar la orden “guardar como”, siendo **archivo** el nombre donde se guardará el documento.
- **:wq** guarda los cambios y sale de *vi*.



```
fp20alu27 — vi programa1.c — 80x24
Es el modo por defecto de vi cuando se abre. Las teclas presionadas ejecutan div
ersas acciones predeterminadas y no se puede editar el texto libremente. Los com
andos son sensitivos a las mayúsculas y a las minúsculas. Algunos ejemplos son:
● ↑ o k mueve el cursor hacia arriba.
● ↓ o j mueve el cursor hacia abajo.
● ← o h mueve el cursor hacia la izquierda.
● → o l mueve el cursor hacia la derecha.
● 1G lleva el cursor al comienzo de la primera línea.
● G lleva el cursor al comienzo de la última línea.
● x borra el carácter marcado por el cursor.
● dd borra o corta la línea donde está el cursor.
● n dd donde n es la cantidad de líneas que se borrarán o cortarán después del
cursor.
● D borra o corta desde la posición de cursor hasta el final de la línea.
● dw borra o corta desde la posición del cursor hasta el final de una palabra.
● yy copia la línea donde está el cursor.
● p pega un contenido copiado o borrado.
● u deshace el último cambio.
~
~
~
~
~
"programa1.c" 16L, 1026B written
```

Figura 1.4 - Información del programa.

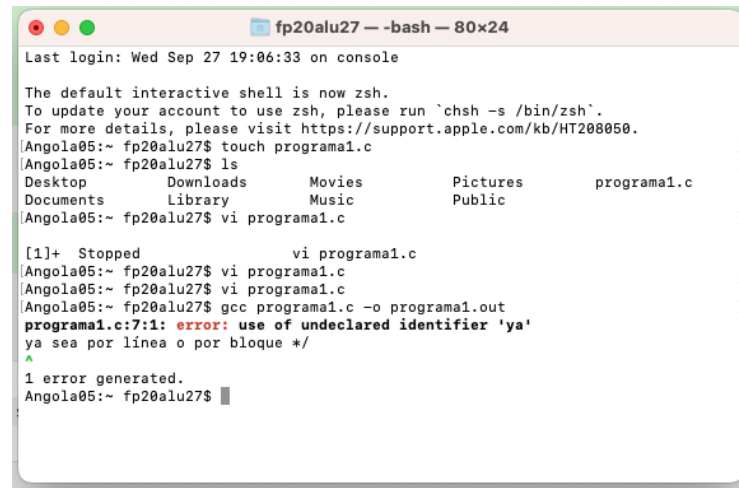


```
fp20alu27 — vi programa1.c — 80x24
Es el modo por defecto de vi cuando se abre. Las teclas presionadas ejecutan div
ersas acciones predeterminadas y no se puede editar el texto libremente. Los com
andos son sensitivos a las mayúsculas y a las minúsculas. Algunos ejemplos son:
● ↑ o k mueve el cursor hacia arriba.
● ↓ o j mueve el cursor hacia abajo.
● ← o h mueve el cursor hacia la izquierda.
● → o l mueve el cursor hacia la derecha.
● 1G lleva el cursor al comienzo de la primera línea.
● G lleva el cursor al comienzo de la última línea.
● x borra el carácter marcado por el cursor.
● dd borra o corta la línea donde está el cursor.
● n dd donde n es la cantidad de líneas que se borrarán o cortarán después del
cursor.
● D borra o corta desde la posición de cursor hasta el final de la línea.
● dw borra o corta desde la posición del cursor hasta el final de una palabra.
● yy copia la línea donde está el cursor.
● p pega un contenido copiado o borrado.
● u deshace el último cambio.
~
~
~
~
~
:wq
```

Figura 1.5 - Comando :wq.

-Primer programa

programa.c

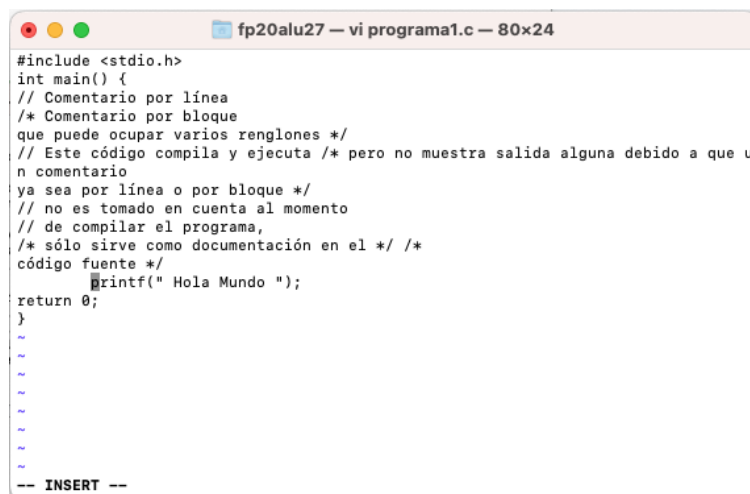


```
fp20alu27 — -bash — 80x24
Last login: Wed Sep 27 19:06:33 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Angola05:~ fp20alu27$ touch programa1.c
Angola05:~ fp20alu27$ ls
Desktop      Downloads    Movies       Pictures     programa1.c
Documents    Library      Music        Public
Angola05:~ fp20alu27$ vi programa1.c

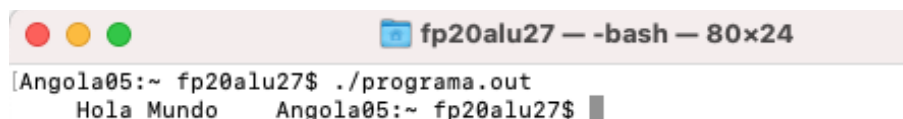
[1]+  Stopped                  vi programa1.c
Angola05:~ fp20alu27$ vi programa1.c
Angola05:~ fp20alu27$ vi programa1.c
Angola05:~ fp20alu27$ gcc programa1.c -o programa1.out
programa1.c:7:1: error: use of undeclared identifier 'ya'
ya sea por línea o por bloque */
^
1 error generated.
Angola05:~ fp20alu27$
```

Figura 1.6 - Terminal, error.



```
fp20alu27 — vi programa1.c — 80x24
#include <stdio.h>
int main() {
    // Comentario por línea
    /* Comentario por bloque
    que puede ocupar varios renglones */
    // Este código compila y ejecuta /* pero no muestra salida alguna debido a que u
    n comentario
    ya sea por línea o por bloque */
    // no es tomado en cuenta al momento
    // de compilar el programa,
    /* sólo sirve como documentación en el */ /*
    código fuente */
    printf(" Hola Mundo ");
    return 0;
}
~
~
~
~
~
~
-- INSERT --
```

Figura 1.7 - Código “Hola mundo”.



```
fp20alu27 — -bash — 80x24
Angola05:~ fp20alu27$ ./programa.out
Hola Mundo  Angola05:~ fp20alu27$
```

Figura 1.8 - Salida “Hola mundo”.

Programa 2.c

```
1 #include <stdio.h>
2
3 //Compiler version gcc 6.3.0
4
5 int main()
6 {
7     short enteroNumero1 = 115;
8     signed int enteroNumero2 = 55;
9     unsigned long enteroNumero3 = 789;
10    char caracterA = 65;
11    char caracterB = "B"
12
13    float puntoFlotanteNumero1 = 89.8;
14
15    unsigned double puntoFlotanteNumero2 = 238.2236;
16    return 0;
17 }
```

Figura 1.9 - Declaración de variables y asignación de valores..

Programa 3.c

```
1 #include <stdio.h>
2
3 //Compiler version gcc 6.3.0
4
5 int main()
6 {
7     //DECLARACIÓN DE VARIABLES
8     int entero;
9     float flotante;
10    double doble;
11    char caracter;
12
13    //ASIGNACIÓN DE VARIABLES
14    entero = 14;
15    flotante = 3.5;
16    doble = 6.8e10;
17    caracter = 'A';
18
19    //FUNCIONES DE SALIDA DE DATOS EN PANTALLA
20    printf("La variable entera tiene valor: %i \n", entero);
21    printf("La variable flotante tiene valor: %f \n", flotante);
22    printf("La variable doble tiene valor: %f \n", doble);
23    printf("La variable caracter tiene valor: %c \n", caracter);
24    printf("Entero como octal: %o \n Como Hexadecimal %X \n", entero, entero);
25    printf("Flotante con precisión: %5.2f \n", flotante);
26    printf("Doble con precisión: %5.2f \n", doble);
27    printf("Carácter como entero: %d \n", caracter);
28
29    return 0;
30 }
```

Figura 1.10 - Programa 3.c.

```
La variable entera tiene valor: 14
La variable flotante tiene valor: 3.500000
La variable doble tiene valor: 68000000000.000000
La variable caracter tiene valor: A
Entero como octal: 16
Como Hexadecimal E
Flotante con precisión: 3.50
Doble con precisión: 68000000000.00
Carácter como entero: 65

Process finished.
```

Figura 1.11 - Salida programa3.c

Programa 4.c

```
1 #include <stdio.h>
2
3 //Compiler version gcc 6.3.0
4
5 int main()
6 {
7     int entero;
8     float flotante;
9
10    printf("Ingresa el valor entero: ");
11    scanf("%i", &entero);
12    printf("El valor ingresado es: %d\n", entero);
13
14    printf("Ingresa el valor float: ");
15    scanf("%f", &flotante);
16    printf("El valor ingresado es: %f\n", flotante);
17
18    return 0;
19 }
20 }
```

Figura 1.12 - Comandos printf y scanf.

```
Ingresa el valor entero: 5
El valor ingresado es: 5
Ingresa el valor float: 4.6
El valor ingresado es: 4.600000
Process finished.
```

Figura 1.13 - Salida programa 4.c

Tarea

1.- Indicar qué sucede cuando en una variable tipo carácter se emplea el formato %d, %i, %o, %x

Se utiliza el porcentaje (%) acompañado del comando "scanf", con el cual se le indica a la computadora el tipo de variable que recibirá, si es del tipo: int, float, char. En el caso de %x y %o, es para expresar el valor de entrada en valores de la forma hexadecimal y octal.

2.- Mencionar las características con las que debe crearse una variable

Se tiene que analizar qué tipo de variable será, ya que al inicio de cada programa se debe especificar que tipo de información maneja, si será del tipo: entero, flotante(números reales), carácter o cadena de caracteres. Cada variable debe tener una escritura diferente a las palabras reservadas usadas en el lenguaje de programación, al igual que su tipo. Con ello posteriormente se puede definir con su tipo correspondiente o tome su valor en el proceso.

3.- Comparación entre Editor de Texto y Procesador de Texto(Realizar una tabla comparativa)

Editor de texto	Procesador de texto
<ul style="list-style-type: none">• Archivos planos.• No tiene formato.• Únicamente sirve para guardar texto, no acepta imágenes.• Cada letra ocupa 1 byte.• El tamaño depende del número de caracteres introducidos.• Terminación .txt• Puede utilizar otras extensiones según el programa lo requiera.	<ul style="list-style-type: none">• Archivos pesados.• Guarda todo tipo de formato.• Se puede modificar cualquier cosa dentro de estos, colores, tipografía y puede incluir imágenes.• Ocupan más espacio, por guardar toda la información relacionada con el formato y los elementos agregados.• Múltiples formatos del más sencillo como .doc y .docx, a los más pesados como .rtf y .odt

4.- Indica los comandos utilizados para compilar y para ejecutar un programa en iOS o Linux

Para compilar: gcc nombrePrograma.c -o nombrePrograma.out; y para la ejecución del programa es: ./nombreprograma.out

5.- Compilación y ejecución de los últimos dos programas de la práctica en DEV C++ u otro IDE en sistema operativo Windows.

Programas desarrollados en visual studio code

-Programa 5.c

```
1 #include <stdio.h>
2 int main() {
3     int enteroNumero;
4     char caracterA = 65; // Convierte el entero a carácter ASCII.
5     double puntoFlotanteNumero;
6     // Asignar valor de teclado a una variable.
7     printf("Escriba un valor entero: ");
8     scanf("%i", &enteroNumero);
9     printf("Escriba un valor real: ");
10    scanf("%lf", &puntoFlotanteNumero);
11    // Imprimir valores con formato.
12    printf("\nImprimiendo las variables enteras \a\n");
13    printf("\t Valor de enteroNumero = %i \a\n", enteroNumero);
14    printf("\t Valor de caracterA = %c \a\n", caracterA);
15    printf("\t Valor de puntoFlotanteNumero = %lf \a\n", puntoFlotanteNumero);
16    printf("\t Valor de enteroNumero en base 16 = %x \a\n", enteroNumero);
17    printf("\t Valor de caracterA en código hexadecimal = %x\n", caracterA);
18    printf("\t Valor de puntoFlotanteNumero\n");
19    printf("en notación científica = %e\n", puntoFlotanteNumero);
20    return 0;
21 }
```

Figura 1.14 - Código programa5.c

```
Escriba un valor entero: 5
Escriba un valor real: 5.6

Imprimiendo las variables enteras
  Valor de enteroNumero = 5
  Valor de caracterA = A
  Valor de puntoFlotanteNumero = 5.600000
  Valor de enteroNumero en base 16 = 5
  Valor de caracterA en código hexadecimal = 41
  Valor de puntoFlotanteNumero
en notación científica = 5.600000e+00

Process finished.
```

Figura 1.15 - Salida programa5.c

-Programa 6.c

```
1 #include <stdio.h>
2 int main(){
3     short ocho, cinco, cuatro, tres, dos, uno;
4
5     // 8 en binario: 0000 0000 0000 1000
6     ocho = 8;
7     // 5 en binario: 0000 0000 0000 0101
8     cinco = 5;
9     // 4 en binario: 0000 0000 0000 0100
10    cuatro = 4;
11    // 3 en binario: 0000 0000 0000 0011
12    tres = 3;
13    // 2 en binario: 0000 0000 0000 0010
14    dos = 2;
15    // 1 en binario: 0000 0000 0000 0001
16    uno = 1;
17    printf("Operadores aritméticos\n");
18    printf("5 modulo 2 = %d\n",cinco%dos);
19    printf("Operadores lógicos\n");
20    printf("8 >> 2 = %d\n",ocho>>dos);
21    printf("8 << 1 = %d\n",ocho<<1);
22    printf("5 & 4 = %d\n",cinco&cuatro);
23    printf("3 | 2 = %d\n",tres|dos);
24
25    printf("\n");
26    return 0;
27 }
```

Figura 1.16 - Código programa6.c

```
Operadores aritméticos
5 modulo 2 = 1
Operadores lógicos
8 >> 2 = 2
8 << 1 = 16
5 & 4 = 4
3 | 2 = 3

Process finished.
|
```

Figura 1.17 - Salida programa6.c

6.- Describe cada una de las palabras reservadas del cuadro mostrado en la práctica con tus propias palabras, se revisa calidad, no cantidad.

<p>auto</p> <p>Tras ser declarada una variable, esta función permite deducir su valor con el valor con el que fue declarado.</p>	<p>double</p> <p>Tipo de variable, relacionada al decimal, con la especialidad de almacenar un mayor rango de números.</p>	<p>int</p> <p>Declarar el tipo de valor que lleva la variable.</p>	<p>struct</p> <p>Estructura, es una colección de variables bajo un nombre, estas variables pueden ser de muchos tipos.</p>
<p>break</p> <p>Rompe el ciclo del programa y manda a finalizar.</p>	<p>else</p> <p>Condición del if, el segundo caso, si es que no se cumple la condición inicial.</p>	<p>long</p> <p>Tipo de variable relacionada al entero, pero almacena un mayor rango de números.</p>	<p>switch</p> <p>Es una estructura condicional múltiple. En que se divide en casos.</p>
<p>case</p> <p>Complemento del switch, separa cada caso/ separación, sin tener que anidar las opciones.</p>	<p>enum</p> <p>Sirve para declarar nuevas numeraciones</p>	<p>register</p> <p>Esto hace que al almacenar una variable, esta sea en el CPU en lugar de la memoria aleatoria.</p>	<p>typedef</p> <p>Permite que el programador defina sus propios datos y crear variables del tipo que se le sea definido.</p>
<p>char</p> <p>Es la serie de caracteres de un tipo</p>	<p>extern</p> <p>Permite reutilizar otra creación de otra ubicación, en lugar una nueva y con una ubicación actual.</p>	<p>return</p> <p>Permite la finalización correcta del programa, evitando crear paradojas.</p>	<p>union</p> <p>Permite almacenar muchos tipos de datos diferentes en la misma región de memoria,</p>
<p>const</p> <p>Es de una variable que tendrá valor y que no puede ser reescrita en procesos subsecuentes.</p>	<p>float</p> <p>Es un tipo de dato que tendrá la variable, en este caso de tipo flotante (número real).</p>	<p>short</p> <p>Es una modificación de rango de valores, en este caso haciéndolo de corto rango.</p>	<p>unsigned</p> <p>Modificación de valores, en este caso el valor no tiene un signo.</p>
<p>continue</p> <p>Salto de instrucción, continuar con.</p>	<p>for</p> <p>Forma iterativa para.,</p>	<p>signed</p> <p>Modificación de valores, en este caso el valor tiene un signo.</p>	<p>void</p> <p>Especifica que la función no devuelve ningún valor.</p>

<p>default</p> <p>Es el caso por defecto, si no se cumple alguna condición o se cumpla, este operador ejecuta las instrucciones.</p>	<p>goto</p> <p>Transfiere el control de una función a una etiqueta, dándole todas sus funciones.</p>	<p>sizeof</p> <p>Es la lectura del peso en memoria</p>	<p>volatile</p> <p>Se usa para indicar al compilador que el valor de la variable puede cambiar inesperadamente.</p>
<p>do</p> <p>Parte de la sentencia do - while, la parte do, se encarga de hacer el bucle(proceso), para pasar a while(condición).</p>	<p>if</p> <p>Condición específica que se comprueba con verdadero y falso, cada una puede contener procesos.</p>	<p>static</p> <p>Donde se almacena la variable, permitiendo que el valor sobreviva aunque este sea modificado.</p>	<p>while</p> <p>Mientras, crea un bucle que ejecuta un proceso, hasta que se cumpla una condición específica</p>

Notas adicionales:

Corrimiento a la izquierda - Al tener el número en binario se hace un corrimiento del bit más significativo a la izquierda

Ejemplo:

8 binario = 0000 0000 0000 1000

Al hacer el corrimiento a la izquierda de 1 bit tenemos: $8 \ll 1$

16 binario = 0000 0000 0001 0000

Al hacer el corrimiento a la izquierda de 2 bits tenemos: $8 \ll 2$

32 binario = 0000 0000 0010 0000

Corrimiento a la derecha - Al tener el número en binario se hace un corrimiento del bit más significativo a la derecha

Ejemplo:

8 binario = 0000 0000 0000 1000

Al hacer el corrimiento a la derecha de 1 bit tenemos: $8 \gg 1$

4 binario = 0000 0000 0000 0100

Al hacer el corrimiento a la izquierda de 2 bits tenemos: $8 \gg 2$

2 binario = 0000 0000 0000 0010

|Conclusión

El alumno obtuvo los conocimientos iniciales para la creación de programas sencillos en el lenguaje C, conoció los elementos de edición, navegación dentro de una terminal de sistema operativo (apple). Ahora puede hacer presentaciones/impresiones de pantalla y pedir datos a terminal, con lo que se empieza a familiarizar en el entorno de desarrollo del lenguaje C; de esta forma ya se concretó el desarrollo secuencial de algoritmos y esto permitirá que el desarrollo de las siguientes prácticas o futuros proyectos tengan ya las bases establecidas en declaración de variables, identificación de los tipos de variables, sus funciones específicas y la forma de su uso.

Referencias

- Mesura, A. (2021). Lenguaje C: características. México: Tecnológico nacional de México. Recuperado el 1 de octubre de 2023:[Palabras Reservadas del lenguaje C - Palabras reservadas del lenguaje c ASIGNATURA : PROGRAMACIÓN - Studocu](#)
- Templos, A.(2022).Manual de prácticas del laboratorio de Fundamentos de programación. México: UNAM.
- S/A. (2018). El lenguaje C. Uruguay: Fing-Udelar. Recuperado el 02 de octubre de 2023:[prinprog-teorico08.pdf \(fing.edu.uy\)](#)