Manejo de la entrada y salida estándar

miércoles, 1 de septiembre de 2021 07:18 a. m.

El lenguaje C ANSI viene acompañado de diversas librerías (directivas) estandarizadas, entre ellas encontramos a <stdio.h>. Dicha directiva contiene funciones que permiten manejar la entrada y salida estándar, las cuales serían teclado y monitor respectivamente.

Las funciones son:

- Funciones de manejo de entrada de datos:
  - getchar()
  - scanf(…)
  - gets()
- Funciones de manejo de salida de datos:
  - putchar(…)
  - printf()
  - puts()

Descripción básica de funciones

miércoles, 1 de septiembre de 2021 07:26 a. m.

Entrada ----> getchar()

Se utiliza para lectura de un solo carácter desde el dispositivo de entrada estándar (teclado, comúnmente). La sintaxis es:

```
char caracter;

caracter= getchar( );  //Llamada a la función getchar() y luego se almacena el carácter leído en la variable
```

Salida ---> putchar()

Usualmente se utiliza para enviar un solo carácter a la salida estándar (comúnmente el monitor) . La sintaxis de su uso son las siguientes:

1.- putchar('h'); //Enviar explícitamente el carácter a visualizar en pantalla. En este caso se envía la "h"

2.- char caracter = '3'; //El carácter se almacena en una variable
putchar(caracter); Se envía a la salida el carácter a través de una variable.

Entrada --> scanf()

La función permite leer un flujo datos, es decir, cualquier combinación alfanumérica: números, cadenas de caracteres, simples caracteres, etc. La función devuelve el número de datos que se recibieron de forma exitosa. Su sintaxis es la siguiente:

```
scanf(cadenaControl, &arg1, &arg2,..., &argN);
```

donde cadenaControl es una referencia de caracteres que especifican el formato de datos identificados; arg1, arg2, ..., argN son los argumentos(direcciones de memoria de las variables donde se almacenaran los datos leídos).

Para el formato de la cadenaControl se debe especificar un carácter '%' junto con un carácter de conversión que indica el tipo de dato a recibir. En la siguiente tabla se describen algunos de estos caracteres de conversión:

cadenaControl	Significado
%c	Se leerá un carácter
%d o %i	Se leerá un entero
%f	Se leerá un flotante
%s	Se leerá una cadena alfanumérica, se detendrá la lectura con un espacio en blanco
%ld	Se leerá un entero tipo long
%lf	Se leerá un flotante de doble precisión
%Regex --> %[A-Z]^.	Se leerá una cadena con especificación Regex=regular expression (expresión regular)

Para el caso de los argumentos(arg1, arg2, ...argN) poder ser variables o inclusive arreglos y sus tipos de dato debe coincidir con lo indicado en la cadenaControl

Cada nombre de la variable debe contener al operador de dirección (&).

Ejemplo:

```
int numero;
float decimal;
char carácter;

fflush(stdin);  //Limpiar buffer de entrada antes de leer, para no leer basura.
scanf("%d %f %c",&numero, &decimal, &caracter );
```

Un cadena se expresa entre comillas dobles (")
Un carácter se expresa entre comillas simples (')

Salida de datos --> printf()

La función nos permite escribir información en la salida estándar (comúnmente el monitor). Permite escribir cadenas alfanumérica, caracteres y valores.

La sintaxis para utilizar printf es la siguiente

printf(cadenaControl, arg1, arg2, …, argN);

donde cadenaControl es un formato en secuencia de caracteres, la cual indica el tipo de los argumentos arg1, arg2, …, argN. Dichos argumentos son los que contienen la información

De igual forma, la cadenaControl se especifica como en la función scanf, es decir, se compone del carácter '%' y de un carácter de conversión especificado en la siguiente tabla:

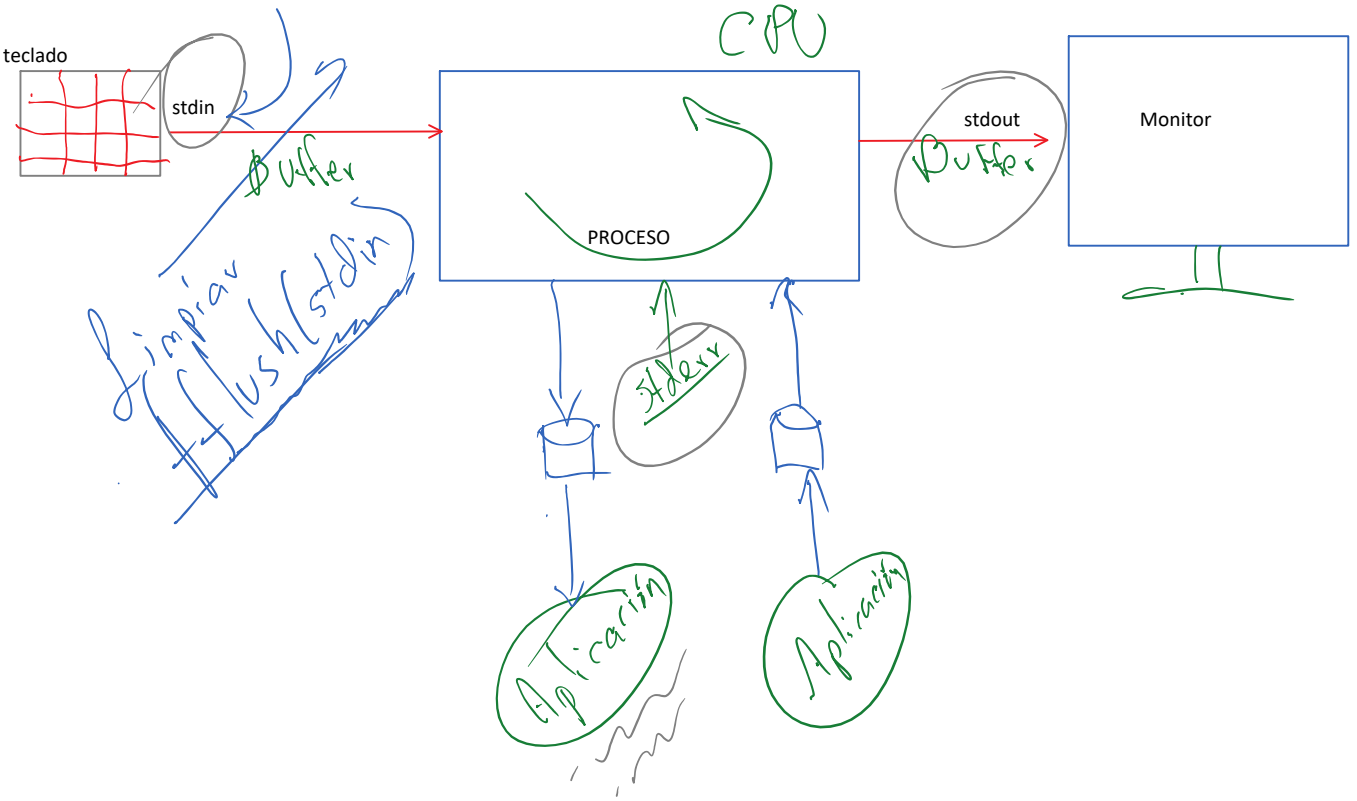
cadenaControl	Significado
%c	Se escribirá un carácter
%d o %i	Se escribirá un entero
%f	Se escribirá un flotante
%s	Se escribirá una cadena alfanumérica.
%ld	Se escribirá un entero tipo long
%lf	Se escribirá un flotante de doble precisión
%1.3lf	Se escribirá un flotante de doble precisión con especificación de cantidad de posiciones enteras y cantidad de decimales después del punto.

Ejemplo:

```
int numero=125;
float costo=12.25;
char letra='U';

printf("%d  %2.2f %c", numero, costo, letra);
```

Proceso de gestión de entrada y salida estándar



Entrada --> gets()
Salida --> puts()

La función puts() permite escribir una cadena de caracteres en la salida estándar; mientras que la función gets() permite leer una cadena de caracteres desde el dispositivo de entrada estándar.

La sintaxis de ambas funciones es la siguiente:

```
puts(<<identificadorCadena>>);
//Ejmplo:
puts("Hola mundo");
```

```
gets(<<identificadorCadena>>);
//Ejemplo
char linea[100]; //Se crea un arreglo vacío para almacenar 100 caracteres
gets(linea);
fflush(stdin);
scanf("%s", &linea);  // Una comparación de gets vs scanf, ambas funciones pueden obtener cadenas
```

Función main

viernes, 3 de septiembre de 2021 07:11 a. m.

Es la función que debemos encontrar en todos nuestros programas. La función marca las operaciones principales del programa.

La función main tiene la siguiente sintaxis:

```
<tipo de dato>   main(){
        //instrucciones secuenciales del programa.
        return <<valor>>;
} ;  Fin programa, fin algoritmo
```

Estructura de un programa en C

Ahora que ya se definió la función principal de un programa, podemos definir a la estructura del código fuente.

```
#include <stdio.h>                    //Sección de librerías o directivas(propias o de terceros)
#define CONSTANTE 3                   //Sección de macros

//Sección de estructuras datos struct & typedef

//Sección de variables globales

//Sección de "prototipos de funciones" e "implementación de funciones".

//Sección de la función main

int main(){

} //Fin del programa

//Aquí se pueden especificar más funciones, siempre y cuando el prototipo de la función se encuentre declarado antes de la función main.
```

COMO COMPILAR EN UNA TERMINAL

En la terminal se debe utilizar el comando cd y ls (DIR) para ubicar el archivo a compilar, deben moverse hasta el directorio donde se encuentra el archivo .c que contiene a la función main.

Ya estando en el directorio que contiene al archivo deben ejecutar el siguiente comando:

```
gcc nombreDelArchivoDeSuPrograma.c
```

Este comando generara (si es que compila a una aplicación con nombre a.exe). Para evitar el nombre a.exe ejecutamos una opción "-o" al compilador

```
gcc nombreDelArchivoDeSuPrograma.c -o nombreMiPrograma
```