Ejercicios de Practica
Lenguaje C

Laboratorio de Computación I
Tecnicatura Superior en Programación.
UTN-FRRQ

Tabla de Contenido

SENTENCIAS DE CONTROL DE FLUJO

1- Decir que muestra cada una de las instrucciones printf.

```
 #include <stdio.h>
#include  <conio.h>
  main( )
        {
         int    expresión, x = 1;
         expresion = ! ( (1 ==  x) &&  4)
         if (expresión )
                  printf("El resultado de la expresión es  verdadero");
        else
                  printf("El resultado de la expresion es falso");
        x = 4;
        for( ; x < 10; x += 3);
                  printf("%d", x / 2);

        x = 10;
         do
        {
                  printf("%d", x--);
         } while ( x >5);
        }
```

2- Incrementar una variable entera **j** desde 0 hasta **n** y luego mostrar el resultado. Hacerlo utilizando 3 bucles diferentes.

3- En la expresión **for(i=1; i==10; i+=2);** cuántas veces se evalúa la condición

4- Leer sucesivamente de teclado, hasta que aparezca un número comprendido entre 1 y 5. Desarrollar el algoritmo usando la función :

a) getchar( ) b) scanf ( )

5- Codificar en C un programa que lea 20 caracteres indique cuantas "a" se ingresaron, cuantas "e, i, o, u"

6- Hacer el algoritmo que imprima los números pares entre 100 y 20 a razón de 6 por línea separados por 3 blancos

7- Hacer un algoritmo que imprima todos los números primos que hay desde la unidad hasta un número que introducimos por teclado. El programa debe poder ejecutarse mientras el usuario lo requiera.

8- Hacer un algoritmo que imprima el mayor y el menor de una serie de 5 números que vamos introduciendo por teclado.

9- Escribir un programa que genere y muestre en pantalla la tabla ASCII.

10- Ingresar un texto de caracteres utilizando la función getchar( ). Indicar la cantidad de caracteres, palabras, y líneas que lo forman. Mostrar lo pedido con carteles aclaratorios. Considerar como separadores de palabras válidos: espacio, tabulador y enter y tener en cuenta que contar palabras no es contar cantidad de separadores. El texto ingresado podrá contener cualquier tipo de caracteres, incluido el enter.

ARREGLOS Y CADENAS- FUNCIONES

11- Dado el siguiente programa indicar que resultado se obtiene.

```
#include<stdio.h>
main()
{
        int vec[]= {1,2,-2,1,3,-1,5,10}, i, var;
        var=1;
        for( i = 0; i <= 7; ++i )
           if(vec[i]>0)
                    var = var * vec[i];
          printf ( " %d", var);
}
```

12- Dada la siguiente declaración e inicialización del arreglo A

```
   {
      int A[5] = { 1,2,3,4,5}
```

Responder:

Valor de A[0] =

Valor de A[4] =

13- Dado el siguiente programa indicar que resultado se obtiene.

```
include<stdio.h>
main()
{
        int vec[]= {1,2,-2,1,3,-1,5,10}, i, var;
        var= i = 0;
        do{
          var = var + vec[i];
          ++i;
         }while (vec[i] >0);
        printf ( " %d", var);}
```

14- Dado el siguiente programa indicar que resultado se obtiene.

```
#include<stdio.h>
main()
{
      int vec[]= {1,2,-2,1,3,-1,5,10,-5,2,3}, i, var;
      var= i = 0;
      while( vec[i] < 10)
      {
         var = var + vec[i];
         ++i;
      }
      printf ( " %d", var);
}
     }
```

15- Escribir un programa que genere aleatoriamente 100 números, los almacene en un arreglo, luego ingrese un número y determine entre los números almacenados cuales son mayores o iguales al número ingresado. Con los mayores calcular su sumatoria y mostrar:

Nº Ingresado

LA SUMATORIA DE LOS NÚMEROS MAYORES QUE **número** ES **sumatoria**

Resultado

16- Escribir un programa que genere aleatoriamente 100 números, los almacene en un arreglo, y luego los muestre ordenados según un indicador de criterio de ordenamiento que debe ingresarse por teclado.

17- Escribir un programa que cree un arreglo de 26 elementos y que además genere y guarde en ellas 26 las letras minúsculas del alfabeto.

18- Escribir un programa que ingrese una cadena, cuente cuantos de sus elementos no son dígitos, cuantos de ellos son blancos y muestre:

Cadena Ingresada

LA CANTIDAD DE NO DÍGITOS QUE APARECEN EN cadena ES cantidad Y L CANTIDAD DE BLANCOS ES cant_blanco

Resultados

19- Escribir un programa que muestre en pantalla lo siguiente:

```
*
**
***
****
*****
```

20- Ingresar dos números enteros y luego presentar el siguiente menú de opciones:

1- SUMAR
2- RESTAR
3- MULTIPLICAR
4- DIVIDIR

A esto el usuario debe responder con la opción correspondiente a la operación que desee realizar entre los números.

- Opción 1: llamar a la función suma, cuyo prototipo es **int suma (int, int);** la función recibe los 2 enteros ingresados y retorna el resultado de la suma para que main lo muestre.
- Opción 2: llamar a la función resta, cuyo prototipo es **void resta (int, int);** la función recibe los 2 enteros ingresados, calcula el resultado de la resta y lo muestra.
- Opción 3: el cálculo y la muestra se realiza en el main
- Opción 4: llamar a la función divi, cuyo prototipo es **float divi (int, int);** la función recibe los 2 enteros ingresados y retorna el resultado exacto de la división para que main lo muestre.

21- Modificar el ejercicio anterior para que se puedan realizar varias operaciones con el par de números ingresados.

22- Modificar el ejercicio anterior para que se puedan ingresar varios pares de números y realizar varias operaciones con cada par de números ingresado.

23- Se dispone de dos cadenas de caracteres **cad1** y **cad2** y se ejecutan las sgtes. funciones:

a. strcpy(cad1, "ANTONIO");
b. strcpy(cad2, "ANTENA");

Para la expresión **strcmp(cad1, cad2);** ¿Cuál de las sgtes. afirmaciones es cierta?

a) Lo que devuelve la expresión dependerá de la dimensión de cad1 y cad2.
b) La expresión devuelve un nº menor que cero.
c) La expresión devuelve un nº mayor que cero.
d) Lo que devuelve la expresión dependerá de la dimensión de cad1.
e) Lo que devuelve la expresión dependerá de la dimensión de cad2.

24- Decir cual será la salida de los siguientes programas.

a)
```
main()
{
char cad1[ ]= "Examen";
char cad2[ ]= "Parcial";
printf("%s\n",strrev(cad2));
printf("%d",strlen(cad1));
}
```

b)
```
main()
  {
        char cadena[20];
        strcpy(cadena, “5, 89, 123,2”);
         strrev(cadena);
         printf ( “ %s \n”, cadena);
        printf ( “ %c \n”, *cadena);
  }
```

25- Hacer un programa que ingrese dos cadenas y:
. Determine cuál de ellas es mayor en orden alfabético.
. Concatene ambas cadenas.
. Determine cuál es mas larga.
. Invierta la primera ingresada.
El programa deberá mostrar todas las salidas con carteles aclaratorios

LA MAYOR ES .....
LAS CADENAS CONCATENADAS QUEDAN.....
LA CADENA MÁS LARGA ES.....
LA CADENA cadena INVERTIDA SE LEE .....

26- Programar una función que, dada una cadena y un carácter, retorne la subcadena que comienza con dicho carácter o NULL en caso de que el caracter no exista en la cadena

27- Ingresar una cadena de caracteres e indicar si la misma es un palíndromo. Realizar esta operación cuantas veces se desee.

28- Dado un texto presentar todas las palabras que lo componen ordenadas alfabéticamente. Optimizar la memoria requerida.

29- Escribir un programa que encuentre el número de veces que una palabra dada ocurre en un texto
Una salida podría ser:
La palabra es "el"
El texto es "el perro, el gato y el canario"
La palabra ocurrió 3 veces.

30- Escribir un programa en **C** que solicite el ingreso de un texto, luego ingrese una letra y muestre, ordenadas alfabéticamente, todas las palabras del texto que comienzan con dicha letra. El ingreso de la letra se podrá repetir tantas veces como se desee. No utilizar variables globales.

(Cant. Max. Caracteres: 1500 – Cant. Max. Palabras: 100 – Long.Max. Palabra: 15 car.)

31- Escriba un programa que inserte un carácter en una determinada posición de una cadena.

32- Cargar un arreglo bidimensional con los nombres de los días de la semana, luego ingresar un número (validar que este entre 1 y 7) y decir que día de la semana le corresponde a ese número. (el programa deberá ser iterativo)

33- Inicializar por programa un arreglo bidimensional con los nombres de los días de la semana, luego ingresar una palabra y decir si la misma es el nombre de un día de la semana. (el programa deberá ser iterativo)

34-

35- Escribir el correspondiente programa en código **C**:
Ingresar por teclado un texto, el final del mismo se dará con **EOF.**
Presentar en pantalla el siguiente menú de opciones:

1- CARACTER
2- COMA
3- LETRA
4- FIN

Y para la opción:

1-CARÁCTER, ingrese un carácter, reemplace los espacios en blanco por el mismo y muestre el nuevo texto.

2-COMA, elimine todas las comas y muestre el nuevo texto.

3-LETRA, ingrese una letra, cuente todas las palabras del texto que empiezan con la misma y muestre apropiadamente el resultado obtenido

4-FIN, finalice el programa.

Nota: El texto se ingresa una única vez, la opción **2** debe ejecutarse sólo una vez y las opciones **1** y **3** deben poder ejecutarse todas las veces que se considere necesario.

.

36- Diseñar una función que cuente cuantas veces aparece cada dígito dentro de un texto.
Su prototipo será: void cuenta(char*, int[], int)
Para ello utilizar un algoritmo prescindiendo de la sentencia switch.

- Diseñar una función que busque y deje disponible al main(no muestre):
  - El dígito con mayor ocurrencia.
  - El dígito con menor ocurrencia.
  - La desviación de la media de ambos valores.

  Su prototipo será: void busca(……)
  Completar

- Incluirlas en un programa escrito en lenguaje C , que ingrese un texto (que pueda contener cualquier carácter) y muestre la información obtenida en busca, y además los dígitos con mayor y menor ocurrencia respectivamente.

NOTA:
- Respetar los prototipos dados.
- No utilizar variables globales

37- Escribir en lenguaje C un programa que solicite el ingreso de un texto (máximo 500 palabras)

El programa deberá mostrar: el texto ingresado, las palabras del texto que se repiten y cuantas veces se repite cada una, con el siguiente formato:

En el texto **"texto"** las palabras que se repiten son:

nnnnnnn	xx veces
nnnnnnn	xx veces
nnnnnnn	xx veces

Una vez ingresado el texto, el mismo deberá ordenarse alfabéticamente por medio de la función **ordena** y luego utilizar la función **muestra** para buscar, contar y mostrar las palabras repetidas. En ambas funciones se exige el uso de las funciones de biblioteca **strcmp(cadena1, cadena2)** y **strcpy(cadena1, cadena2)**

NOTA: * NO SE PERMITE EL USO DE VARIABLES GLOBALES
* UTILIZAR CARTELES ACLARATORIOS PARA EL INGRESO DE DATOS

Por ej., si se ingresó el texto **esto es un examen de sintaxis y sintaxis es una materia de segundo año y es necesario para nosotros aprobarlo,** la salida deberá ser:

En el texto **"esto es un examen de sintaxis y sintaxis es una materia de segundo año y es necesario para nosotros aprobarlo"** las palabras que se repiten son:

de	**2 veces**
es	**3 veces**
sintaxis	**2 veces**
y	**2 veces**

OTROS TIPOS DE DATOS- ESTRUCTURAS

38- Dado el siguiente segmento de programa:

```
struct dato {int codigo;
             char descripcion[20];
              float precio;
                    }a;
main()
        { struct dato b;
    ……
    ……
}
```

a) ¿Son **a** y **b** dos variables que tienen el mismo tipo de dato? **SI - NO**
b) ¿La variable **a** es de ámbito local? **SI - NO**

39- Definir una estructura de nombre **alumnos** para almacenar la siguiente información:

- nombre del alumno
- nº de legajo
- domicilio
- turno,

declarar una variable para almacenar los datos de cada uno de los 300 alumnos de un establecimiento.
Luego de declararla asignar los sgtes. datos en la 1ª estructura: Juan Gomez , 123245, San Juan 1056, M

40- Dada la siguiente declaración:

```
struct asistencia {
        char turno;
         int dia [6];};
```

a) Declarar una variable alumnos de tipo estructura **asistencia**.
b) Asignar al campo turno de la variable alumnos una **M** y un **0** a la primera componente del campo dia.

PUNTEROS

41- Decir cual será la salida:

```
a   #include<stdio.h>
    mainO
        {
         int arr[10] = {23, 5, 98, 65, 3, 55, 73, 9, 21, 85}, *p;
```

```
p = arr;
 printf ( " %d\n", arr[*(p + 7)]);
 printf ( " %d\n", *arr + 3);
 printf ( " %d\n", *p++);
 printf(" %d\n", *(arr + 1));
 printf(" %d\n", (*p)++);
printf(" %d\n", *p);
printf(" %d\n", *p++);
printf(" %d\n", *p);
}
```

42- Suponiendo que **p** es un puntero a float que actualmente apunta a la posición 100 y que los floats tienen 4 bytes de longitud ¿cuál es el valor de P después de que se haya ejecutado la siguiente sentencia?

```
p = p + 2;
```

43- Dado el sgte. Trozo de programa:

```
char u, v = 'A';
char *pu, *pv = &v;
…….
*pv = v + 1;
u = *pv + 1;
pu = &u;
```

y suponiendo que el valor asignado a **u** se almacena en la dirección F8C y el valor asignado a **v** se almacena en la dirección F8D, decir:

a ¿Qué valor es asignado a pv?
b ¿Qué valor es representado por *pv?
c ¿Qué valor es asignado a u?
d ¿Qué valor es asignado a pu?
e ¿Qué valor es representado por *pu?

44- Escribir un programa en C que inicialice un arreglo bidimensional con los números del 1 al 6 y luego los muestre utilizando punteros.

45- Rehacer el ejercicio 1 utilizando los sgtes. prototipos:

```
void suma (int, int, int *);
void divi (int, int, float*);
int * multi (int, int);
int * resta (int, int);
```

suma y **divi** reciben los 2 operandos y la dirección de una variable donde almacenarán el resultado de la operación. El resultado se muestra en main.

multi y **resta** reciben los 2 operandos y retornan el resultado de la operación. El resultado se muestra en main

46- Realizar un programa que me permita ingresar enteros (máximo 20, de lo contrario 0 (cero) indica fin de ingreso) y almacenarlos en un arreglo. Luego programar una función de nombre **busca**, que recibe el arreglo de enteros y devuelve un puntero al primer valor que no esté ordenado en forma creciente. Si el arreglo está ordenado retornará NULL Ej: Si el arreglo es: 44, 58, 65, 13, 33, 0 la función deberá retornar un puntero al cuarto elemento. Luego el main deberá mostrar: El 13 esta desordenado y es el elemento nro. 4 del arreglo. El prototipo de la función busca es:

```
int * busca (int [ ]);
```

Resolverlo utilizando aritmética de punteros (no utilizar subíndices).

47- Hacer un programa que permita ingresar la información de los alumnos de una determinada institución (Máximo 100 alumnos). Luego presente en pantalla el sgte. menú:

1- ORDENA
2- BUSCA LEGAJO
3- SALIR

Los datos se almacenaran con el sgte formato:

```
struct fecha
{
     int dia, mes, anio;
};
struct slumnos
{
     char legajo [10];
     char ap-nom [35];
     char direc [40];
     struct fecha fe [2];  /* fecha nacimiento y fecha ingreso institución */
     int edad;
};
```

y se utilizaran las sgtes. funciones:

void carga_alu (struct alumnos [ ], int *); Recibe el arreglo de estructuras donde se almacenaran los datos y la dirección de un entero donde se almacenará la cantidad de alumnos cargados

struct alumnos * ordena_ap (struct alumnos [ ], int); Recibe los datos de todos los alumnos y la cantidad de alumnos cargados y retorna la dirección de un nuevo arreglo de estructuras que contendrá todos los alumnos ordenados por apellido. El main mostrara el listado de los alumnos ordenado por apellido.

struct alumnos * busca_leg (struct alumnos [ ], char *, int); Recibe los datos de todos los alumnos, la cantidad de alumnos cargados y un legajo, retorna la dirección de la estructura que contiene los datos del alumno cuyo legajo coincide con el buscado. En caso de no encontrarlo retornará la dirección de una estructura que contendrá "*********" (5 asteriscos) en el campo legajo. El main mostrará los datos del alumno encontrado o **El legajo no existe** en caso contrario.

48- Desarrollar un programa en lenguaje C, que solicite el ingreso de un texto, luego presente el siguiente menú en pantalla:

1. Cuenta vocales
2. Cuenta consonantes
3. Salir

A continuación el main mostrará la cantidad de vocales y/o consonantes, según corresponda, que hay en el texto ingresado. El programa deberá ser iterativo.

Para resolver esta problemática deberá utilizar las siguientes funciones:

- Una función que cargue un texto (que incluya cualquier tipo de caracteres, incluso la nueva línea) cuyo prototipo será: **char * carga (void);**

- Una función que solamente determine si un carácter es una vocal (no cuente cantidad de vocales), su prototipo será: **void esVocal(char, int*)**

- Una función que solamente determine si un carácter es una consonante (no cuente cantidad de consonantes), su prototipo: **int esConsonante(char)**
  Tener en cuenta que una consonante es un carácter alfabético que NO es una vocal.

49- Escribir un programa C que:
Permita ingresar un texto, el cual podrá contener cualquier tipo de caracteres, incluido el enter (nueva línea o \n) (máximo 1000 caracteres) utilizando la función **carga**, cuyo prototipo es:

char * carga (int *)

Luego muestra el siguiente menú:

1- Modifica
2- Cuenta
3- Busca
4- Salir

Opción 1: solicitar al usuario que ingrese dos caracteres, un carácter a reemplazar y un carácter de reemplazo, a continuación llamar a la función **modifica**, la que deberá devolverle al main la dirección de una estructura que contenga:

- El texto original
- El texto modificado
- La cantidad de reemplazos que se realizaron

para que la función main imprima dicha información. El prototipo de **modifica** es:

struct modif * **modifica** (char *, int, char, char);

Opción 2: solicitar al usuario que ingrese un caracter y llamar a la función **cuenta**, la que deberá contar y dejar disponible para el main la cantidad de veces que dicho caracter aparece en el texto. El prototipo de **cuenta** es: void **cuenta** (char *, int, char, int *)

Opción 3: llamar a la función **busca**, esta función solicita al usuario el ingreso de un caracter y retorna la dirección de memoria en la que se encuentra la primera ocurrencia dentro del texto del carácter ingresado o NULL si no lo encuentra, a continuación la función main imprimirá el texto a partir de la dirección retornada o "el carácter no existe en el texto" en caso de no encontrarlo. El prototipo de **busca** es:
char * busca (char *, int)

Opción 4: finaliza el programa

50- Realizar un programa en C que permita ingresar palabras, las busque en un mini diccionario y muestre su significado y clasificación gramatical. Este proceso deberá ser iterativo.
Las palabras se almacenaran mediante la función

struct dicc * ingreso_minidicc (int * canti_pal);

en un arreglo de estructuras que tendrán el siguiente formato:

```
struct dicc {
   char pal[20];
   char significado[50];
   char clasificación[20];
};
```

La búsqueda se realizará a través de la función

struct dicc * busca(**struct dicc. [ ], int canti_pal, char * pal**);

si la misma es satisfactoria retornará la dirección de la estructura que contiene la palabra ingresada, caso contario retorna la dirección de una estructura que contiene "*****" en el campo **significado**. En este último caso se tendrá que recurrir a la función **ortografía** cuyo prototipo es:

char * ortografia (struct dicc [], int canti_pal, char pal_p_corregir[]);

Esta función muestra en pantalla un listado de las palabras que figuran en el mini diccionario que tengan las dos primeras letras iguales a las de la palabra no encontrada, una vez seleccionada la palabra correcta la retorna a main para buscarla nuevamente y ver su significado y clasificación gramatical en pantalla.