TRABAJO PRÁCTICO de LABORATORIO N°1

Códigos y Límites

Objetivos:
Con la realización del siguiente trabajo práctico se busca:

- Conocer algunos de los diversos límites que tiene el computador.
- Identificar desde la programación, donde se hallan esos límites.
- Conocer como estos límites se relacionan con los códigos usados internamente por el computador.
- Ver cómo, aprovechando el conocimiento de los códigos, se puede mejorar la programación.

Introducción de contexto:

Cuando se desarrolla un programa, con independencia del lenguaje usado, raramente se cuestiona la exactitud de los resultados obtenidos. Al notar algún error en los resultados se revisa el programa en su estructura, código, etc. pero difícilmente se piensa en la posibilidad de haber superado algún límite impuesto por el computador o por la forma interna de codificar.

Este trabajo práctico a punta a encontrar la forma de poner de manifiesto estos límites, y relacionarlos con los códigos internos que usa el computador.

Se suministra programas en lenguaje C y algunas funciones de lenguaje Python para verificar el tratamiento de los distintos tipos de datos en memoria y sus límites.

Elementos Necesarios:

- Una computadora en condiciones de ejecutar un compilador de lenguaje “C”
- Editor de texto adecuado. Compilador de lenguaje C (opcional) ó Compiladores WEB de lenguaje C.
- Interprete Python 3.6 o superior o Interprete WEB de Python 3.XX
- Conocimiento básico de programación en lenguaje C y Python.
- Acceso a Internet.

Nota:

- La guía contiene los programas fuente y una copia de la salida despues de su ejecución. (Párrafos de este documento en color azul)
- Permite ejecutar los programas usando las opciones sugeridas en el anexo II
- Se puede desarrollar las conclusiones sin necesidad de ejecutar los programas.
- La ejecuciones sobre WEB no reflejan la arquitectura de la máquina utilizada.

Desarrollo del Trabajo Práctico:

1. **Codificación de caracteres alfanuméricos:** Utilizando el block de notas del Windows®, **NotePad** se escribe el siguiente texto:

   El año está por comenzar

   Luego se lo guarda en el disco en un directorio de trabajo accesible:

   archivo1.txt formato ANSI
   archivo2.txt formato UTF-8 (codigo de longitud variable 8 y 16 bits)
   archivo3.txt formato UTF-16B (código de 16 bits Big Endian)
   archivo4.txt formato UTF-16L (código de 16 bits Little Endian)

   Mediante el intérprete de comandos **(cmd)** posicionado en el directorio que contiene los archivos, se procede a visualizar el tamaño de cada archivo utilizando el comando **dir** y luego aplicar el comando **type** sobre cada archivo

```
C:\Laboratorio-1>dir

08/09/2020  06:36 p.m.    <DIR>          .
08/09/2020  06:36 p.m.    <DIR>          ..
08/09/2020  06:35 p.m.                24 archivo1.txt
08/09/2020  06:35 p.m.                29 archivo2.txt
08/09/2020  06:36 p.m.                50 archivo3.txt
08/09/2020  06:36 p.m.                50 archivo4.txt
               4 archivos            153 bytes

C:\Laboratorio-1>type archivo1.txt
El a±o estß por comenzar

C:\Laboratorio-1>type archivo2.txt
´╗┐El a├▒o est├í por comenzar

C:\Laboratorio-1>type archivo3.txt
■ E l   a ± o   e s t ß   p o r   c o m e n z a r

C:\Laboratorio-1>type archivo4.txt
El año está por comenzar
```

   Explique y justifique a qué se debe la diferencia entre el texto escrito original y el visualizado usando el comando del SO Windows ***type.***

2. Ejecute el programa **LAB1-2.C** permite ingresar una letra y la cambia por su equivalente en mayúscula o minúscula según sea el ingreso efectuado.

```
Ingrese una letra: A
La letra igresada fue A, ahora es a
Presione una tecla para continuar . . .

Ingrese una letra: z
La letra igresada fue z, ahora es Z
Presione una tecla para continuar . . .

Ingrese una letra: ñ
El caracter Ingresado esta fuera de rango
Presione una tecla para continuar . . .

Ingrese una letra: 3
El caracter Ingresado esta fuera de rango
Presione una tecla para continuar . . .
```

- Analizar el método de cambio utilizado
- Observar y analizar la validación de rangos
- Explicar como se podría extender la funcionalidad del programa para procesar caracteres del lenguaje español **á, é, í, ó, ú, ü, ñ** y sus mayúsculas.
  a) Para codificación ***ANSI***
  b) Para codificación ***EASCII***

3. **<u>Medición del tamaño ocupado en memoria por los tipos de datos en Bytes:</u>** Ejecute el programa **LAB1-3.C**, para determinar el tamaño en Bytes de los tipos de datos incluídos en el lenguaje : *(datatype)*

- ***char***
- ***int***
- ***short int***
- ***long int***
- ***long long int***
- ***unsigned int***
- ***float***
- ***double***

```
ESPACIO OCUPADO POR LOS TIPOS DE DATOS EN C

CARACTERES ALFANUMERICOS:

"char"..............: 1 Byte

NUMEROS ENTEROS:

"int"...............: 4 Byte
"short int".........: 2 Byte
"long int"..........: 4 Byte
"long long int".....: 8 Byte
"unsigned int"......: 4 Byte

NUMEROS REALES:

"float".............: 4 Byte
"double"............: 8 Byte

Enter para terminar:
```

Indique el compilador que utiliza para realizar la prueba, tanto su nombre comercial como el fabricante y la versión, tambien especifique en qué procesador se realizó la prueba. En caso de utilizar trabajo sobre web indicar la página que proporciona el servicio.
Nota: En lenguaje C/C++/C# se utiliza la función interna ***sizeof (****datatype****)****.*

4. **Medición del rango de un número entero:** El programa **LAB1-4.C** incrementa tres variables numéricas declaradas como entero ***(int)***, entero sin signo ***(unsigned int)*** y entero corto ***(short int)***. En las tres se incrementa su valor hasta llegar al Valor Máximo positivo (***Vmax***) y se le suma 1 (***Vmax +1***).
Nota: Se omitieron otras declaraciones pero funciona con (long long int) y (char), este último se utiliza como carácter pero tambien puede usarse como entero de ocho bits en operaciones numéricas)

```
Calculando...

Para int          :  Vmax es 2147483647 y Vmax+1 es: -2147483648

Para unsigned int :  Vmax es 4294967295 y Vmax+1 es:           0

Para short int    :  Vmax es      32767 y Vmax+1 es:      -32768

Enter para terminar:
```

Pruebe y justifique los resultados del programa ¿Explique cuál es la razón por la cual se obtuvo el valor de **2147483647; 4294967295 y 32767** ? Explique porqué se obtienen los valores negativos al sumarles 1.Dermine los valores que se obtendrían con (long long int) y (char) no usar en este caso el modificador unsigned

5. **Visualización de la codificación de los números reales usando el tipo (float):** Al ejecutar el programa LAB1-5.C , se solicita el ingreso de números reales y vuelca el contenido almacenado en memoria. Se solicita verificar maualmente los resultados para los siguientes números
   Nota: LSB significa que de los cuatro BYTES mostados el de la derecha corresponde al los dígitos menos significativos de los 32 utilizados

   a) **105.67**
   b) **-1.0**
   c) **0.0**
   d) **-16.1**
   e) **12.75**
   f) **32.0**

```
PF IEEE-754 en simple precision ocupa : 4 Bytes
Ingrese un numero real (numero mayor a 999. Termina): 105.67

Valor ingresado     : 105.669998
Valor en hexadecimal: 42 D3 57 0A LSB

Ingrese un numero real (numero mayor a 999. Termina): 0.0

Valor ingresado     : 0.000000
Valor en hexadecimal: 00 00 00 00 LSB

Ingrese un numero real (numero mayor a 999. Termina): -16.1

Valor ingresado     : -16.100000
Valor en hexadecimal: C1 80 CC CD LSB

Ingrese un numero real (numero mayor a 999. Termina): 12.75

Valor ingresado     : 12.750000
Valor en hexadecimal: 41 4C 00 00 LSB

Ingrese un numero real (numero mayor a 999. Termina): 32.0

Valor ingresado     : 32.000000
Valor en hexadecimal: 42 00 00 00 LSB
Ingrese un numero real (numero mayor a 999. Termina): 9999
```

6. **Medición del ordenamiento de Bytes de un computador:** Ejecutar el programa **LAB1-6.C**, que permite detectar si el equipo en el cual se ejecuta es del tipo “big endian”, “little endian” o "middle endian”, Expliacar que inconvenientes pueden acarrear no tomar en cuenta el ordenamiento de Bytes (***Endianess***)

```
Este microprocesador es LITTLE ENDIAN

Enter para salir:
```

7. **Mostrar la posición de memoria usada por una variable:** Ejecutar el programa **LAB1-7.C**, que permite mostrar las direcciones de memoria donde fue guardado cada uno de los bytes. Se ingresó el número decimal ***305419896*** que en Hexadecimal usando 32 bits resulta ***0x12345678*** Explicar como se interpreta el ordenamiento de bytes y decir si el resultado obtenido refleja el uso de un procesador big, lttle o middle endian.,

```
La codificacion de dato tipo 'int' ocupa 4 Bytes
Tenga presente el rango para los  Bytes ocupados

Ingrese una variable entera: 305419896


Recuede que los valores negativos estan en Ca2
Ordenamiento: Big Endian    Byte 0 (MSB)
Ordenamiento: Little Endian Byte 0 (MSB)

Direccion de memoria de la variable:0x0240FF24

Posicion: 0x0240FF24 - contenido Byte 0 : 0x78
Posicion: 0x0240FF25 - contenido Byte 1 : 0x56
Posicion: 0x0240FF26 - contenido Byte 2 : 0x34
Posicion: 0x0240FF27 - contenido Byte 3 : 0x12

Enter para salir:
```

ANEXO I

FUENTES DE PROGRAMAS EN C

Programa LAB1-2.C

```
/*-PROGRAMA: LAB1-2.C-2020-*/
#include <stdio.h>
#include <stdlib.h>
int main (void)
{
      char letra1, letra2;
      letra1=0;
      letra2=0;
      printf ("Ingrese una letra: ");
      letra1=getchar ();
      if (letra1>='A' && letra1<='Z')
            letra2=letra1+('a'-'A');
      if (letra1>='a' && letra1<='z')
            letra2=letra1-('a'-'A');
      if (letra1<'A' || (letra1>'Z' && letra1<'a') || letra1>'z')
            printf ("El caracter Ingresado esta fuera de rango\n");
      else
            printf ("La letra era %c, ahora es %c \n", letra1, letra2);
     system ("pause");
      return 0;
}
```

Programa LAB1-3.C

```
/*-PROGRAMA: LAB1-3.C-2020-*/

#include<stdio.h>

int main()

{

      printf("\nESPACIO OCUPADO POR LOS TIPOS DE DATOS EN C\n\n");

      printf("\nCARACTERES ALFANUMERICOS:\n\n");

      printf("\"char\".............: %d Byte\n\n",sizeof(char));

      printf("NUMEROS ENTEROS:\n\n");

      printf("\"int\"..............: %d Byte\n",sizeof(int));

      printf("\"short int\".........: %d Byte\n",sizeof(short int));

      printf("\"long int\"..........: %d Byte\n",sizeof(long int));

      printf("\"long long int\"......: %d Byte\n",sizeof(long long int));

      printf("\"unsigned int\".......: %d Byte\n\n",sizeof(unsigned int));

      printf("NUMEROS REALES:\n\n");

      printf("\"float\"............: %d Byte\n",sizeof(float));

      printf("\"double\"...........: %d Byte\n\n",sizeof(double));

      printf("Enter para terminar:\n");

      system("pause >nul");

}
```

Programa LAB1-4.C

```
/*-PROGRAMA: LAB1-4.C-2020-*/

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

int main (void)

{
     int a,i;
     unsigned int b,j;
     short int c,k;
     a=0;b=0;c=0;
     i=1;j=1;k=1;

printf ("\nCalculando...\n\n");

     while ( i >=0 )
          {
          a++;
          i++;
          }

     while ( j >0 )
          {
          b++;
          j++;
          }

     while ( k >=0 )
          {
          c++;
          k++;
          }

printf("Para int          : Vmax es %d y Vmax+1 es: %d\n\n",a,i);
printf("Para unsigned int : Vmax es %u y Vmax+1 es:          %u\n\n",b,j);
printf ("Para short int   : Vmax es      %d y Vmax+1 es:     %d\n\n",c,k);
printf ("Enter para terminar:\n");
system ("Pause >nul");
return 0;

}
```

Programa LAB1-5.C

```
/*-PROGRAMA: LAB1-5.C-2020-*/

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

int main()

{

     float nroReal;
     unsigned char *lowByteMem=(unsigned char*)&nroReal;
     int i,
     tam=sizeof(float);

printf("\nPF IEEE-754 en simple precision ocupa: %d Bytes\n\n", sizeof(float));
printf("Ingrese un numero real (numero mayor a 999. Termina): ")
scanf("%f", &nroReal);
while (nroReal < 999.)

            {

            printf("\nValor ingresado     : %f\n", nroReal);
            printf("Valor en hexadecimal: ");
            /*muestra los bytes donde está almacenado el número XX XX XX XX LSB*/
            i=tam-1;

            while (i >=0)

               {
               printf("%02X ", lowByteMem[i]);
               i = i - 1;
               }

            printf("LSB");
            printf("\n\n");
            printf("Ingrese un numero real (numero mayor a 999. Termina): ");
            scanf("%f", &nroReal);

            }
return 0;
}
```

Programa LAB1-6.C

```
/*-PROGRAMA: LAB1-6.C-2020-*/
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        int i = 0x12345678;
        unsigned char * pi = (unsigned char *) &i;
        if(pi[0]==0x12&&pi[1]==0x34&&pi[2]==0x56&&pi[3]==0x78)
             {
             printf("\nEste microprocesador es BIG ENDIAN\n\n");
             }
        else if(pi[0]==0x78&&pi[1]==0x56&&pi[2]==0x34&&pi[3]==0x12)
             {
             printf("\nEste microprocesador es LITTLE ENDIAN\n\n");
             }
        else
             {
             printf("\nEste microprocesador es MIDDLE ENDIAN\n\n");
             }
        printf("Enter para salir:\n\n");
        system ("pause >nul");
        return 0;
}
```

Programa LAB1-7.C

```
/*-PROGRAMA: LAB1-7.C-2020-*/

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


int main()
{
  int i;
  unsigned char * pc = (unsigned char *) &i;

  printf("\nLa codificacion de dato tipo 'int' ocupa %d Bytes\n",sizeof(i));
  printf("Tenga presente el rango para los  Bytes ocupados\n\n");
  printf("Ingrese una variable entera: " );

  scanf ("%d", &i);

  printf("\n\nRecuede que los valores negativos estan en Ca2\n");
  printf("Ordenamiento: Big Endian    Byte 0 (MSB)\n");
  printf("Ordenamiento: Little Endian Byte 0 (MSB)\n\n");
  printf("Direccion de memoria de la variable:0x%p\n\n",&i);

  printf("Posicion: 0x%p - contenido Byte 0 : 0x%02X\n",pc+0, pc[0]);
  printf("Posicion: 0x%p - contenido Byte 1 : 0x%02X\n",pc+1, pc[1]);
  printf("Posicion: 0x%p - contenido Byte 2 : 0x%02X\n",pc+2, pc[2]);
  printf("Posicion: 0x%p - contenido Byte 3 : 0x%02X\n\n",pc+3, pc[3]);

  printf("Enter para salir: ");
  system("Pause >nul");
  return 0;
}
```

ANEXO II

FUNCIONES PYTHON 3.XX RELACIONADAS CON EL TP.

Python 3.6.5 on win32

```
'CONVERSIONES A DECIMAL'

>>> print (0xABCD)
43981
>>>
>>> print (0o1234)
668
>>>
>>> print (0b1010)
10

'CONVERSIONES DESDE DECIMAL'

>>> print(hex(1234))
0x4d2
>>>
>>> print(oct(1234))
0o2322
>>>
>>> print(bin(1234))
0b10011010010

'CONVERSION DE UNA CADENA EN CUALQUIER BASE A NUMERO DECIMAL'

>>> int('101010',2)
42
>>> int('1212',3)
50
>>> int('1231',4)
109
>>> int('1234',5)
194
>>> int('1357',8)
751
>>> int('1479',10)
1479
>>> int('1AC0',16)
6848
>>>
```

```
'CONVERSION DE NUMERO A CADENA'

>>> str(1234)
'1234'
>>>

'CONVERSION DE CADENA A NUMERO'

>>> int('1234')
1234
>>>

'CONVERSION DE EL ORDEN DECIMAL DE LA TABLA ASCII A CARACTER'

>>> print (chr(65))
A
>>> print (chr(65),chr(66),chr(67))
A B C
>>>

'CONVERSION DE UN CARACTER AL ORDEN DECIMAL DE LA TABLA ASCII'

>>> print (ord('A'))
65
>>>
```