

Tarea 2. Programación y Algoritmos

Carlos Giovanni Encinia González

agosto 2021

Ejercicio 1

a) ¿Cuál es la diferencia entre el format `%i` y `%d`? Dé un ejemplo.

b) ¿Cuál es la diferencia entre la declaración `bool` y `_Bool` en C?

Respuesta

a)

Para la función `printf()` el parámetro `%i` produce el mismo resultado que `%d` imprimen un entero en base decimal, lo interesante es lo que sucede en la función `scanf()`.

Cuando utilizamos el formato `%i`, C nos da la posibilidad de poder ingresar números en distintas bases, por defecto lee los números en base decimal y si queremos que el número dado se lea en otra base, tenemos que especificar dicha base de la siguiente manera.

Si la base es hexadecimal escribimos al inicio `0x[numero]`.

Si la base es octal escribimos `0[numero]`.

Si imprimimos los números leídos en esas bases, dará la equivalencia en la base decimal. En seguida se muestran los resultados de la implementación del código en C (1).

Para la figura (1) se ingresa un número en base hexadecimal y se regresa su equivalencia en decimal $17_{16} = 23_{10}$.

Para la figura (2) se ingresa un número en base octal y se regresa su equivalencia en decimal $55_8 = 45_{10}$.

Figure 1: Hexadecimal

```
Escribe un numero para el formato i
0x17
El numero con formato d es: 23
El numero con formato i es: 23

Escribe un numero para el formato d
17
El numero con formato d es: 17
El numero con formato i es: 17
```

Figure 2: Octal

```
Escribe un numero para el formato i
055
El numero con formato d es: 45
El numero con formato i es: 45

Escribe un numero para el formato d
055
El numero con formato d es: 55
El numero con formato i es: 55
```

b)

La principal diferencia es que la declarativa `_Bool` se utiliza sin tener que llamar a alguna librería y la declarativa `bool`, debe ser usada con la librería `#include <stdbool.h>`, si se crea una variable con el valor 0 esta se tomara como false en condicionales y ciclos while mientras que cualquier otro número diferente de 0 tomara un true.

Figure 3: _Bool

```
_Bool 0 es falso
_Bool != 0 es verdad
```

Si no utilizamos la librería necesaria para `bool` nos mostrara el compilador el siguiente error:

Error : unknown type name 'bool'

Code 1: Problema 1

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

void print_dif(char type)
{
    int number;

    printf(\
"Escribe un numero para el formato %c\n"\
        , type);
    if(type == 'i')
    {
        scanf("%i", &number);
    }
    else
    {
        scanf("%d", &number);
    }

    printf(\
"El numero con formato d es: %d\n"\
        , number);
    printf(\
"El numero con formato i es: %i\n\n"\
        , number);
}

int main()
{
    char i='i', d='d';
    _Bool verdad = 2, falso = 0;
    bool hola=1;

    print_dif(i);
    print_dif(d);

    printf("_Bool 0 es %s\n"\
        , falso?"verdad":"falso");
    printf("_Bool != 0 es %s\n"\
        , verdad?"verdad":"falso");

    return 0;
}
```

Ejercicio 2

a) ¿Qué pasa si al leer un entero con `scanf()`, el usuario teclea el número seguido con una letra? Ejem: 67f¿como explica el resultado?

b) Enseguida de esta instrucción, añade ahora la lectura de un carácter, ¿Qué pasa y como explica este comportamiento?

Respuesta

a) Si lo que se ingresa en pantalla es un numero seguido de una letra y el formato en `scanf()` es para enteros, lo que hará es imprimir solamente el numero. Esto sucede ya que C lee únicamente el tipo de dato que se puede tomar como el formato lo indica, y lo que queda del dato ingresado se almacena en el buffer.

Figure 4: number character

```
Ingresa un numero con una letra al final
89g
El numero es: 89
```

b) Si en seguida de esto se hace una lectura de caracteres, debido a que en el buffer se quedó la letra, lo que hará el `scanf()` será leer lo que hay en el buffer es decir la letra, entonces esto es lo que se guardara en la variable en el `scanf()` y no esperará a que se ingrese algún dato ya que terminó de ejecutarse la instrucción.

Figure 5: Desaparece letra

```
Ingresa un numero con una letra al final
89g
El numero es: 89
```

Figure 6: New result

```
Ingresa un numero con una letra al final
1024b
El numero es: 1024
letra: b
```

Code 2: Problema 2

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

```
int main()
{
    int number_i, n;
    float number_f;
    char ae;

    printf("Ingresa un numero\
con una letra al final\n");
    scanf("%d", &number_i);
    printf("El numero es: %d\n", \
        number_i);
    scanf("%c", &ae);
    printf("letra: %c", ae);
    return 0;
}
```

Ejercicio 3

Programa que realice una operación aritmética especificada entre dos fracciones. La entrada debe ser de la forma: $a/b \odot c/b$, donde $\odot \in \{+, -, *, /\}$.

Respuesta

Se creó un programa código (3) que utiliza `scanf()` y alguna de sus funcionalidades, como input se le da una serie de datos que deben tener la forma $\frac{a}{b} \odot \frac{c}{b}$ donde $a, b, c \in \mathbb{N}$, dando así el resultado de dicha operación, es importante destacar que los denominadores de ambas fracciones deben de ser iguales, si esto no es así el programa termina diciendo que los datos que se dieron no son correctos. En seguida se muestran algunos de los resultados obtenidos durante la ejecución del algoritmo.

Figure 7: suma

```
Ingresa operacion fracciones (+, -, *, /)
a/b o c/b
4/2+8/2
=====
4/2+8/2 = 12/2
=====
```

Code 3: Problema 3

```
//Giovanny Encinia
// 23-08-2021
#include <stdio.h>
#include <stdlib.h>

void resultado(int a, int b, char operation, int c, int d, \
               int resul, int denom)
{
    /*Funcion para imprimir el resultado*/

    printf("=====\\n\\n");
    printf("          %d/%d%c%d/%d = %d/%d\\n\\n\\n" \
           , a, b, operation, c, d, resul, denom);
    printf("=====\\n");
}

int main(void)
{
    int a, b, c, d;
    char div_1, operation, div_2;

    printf("Ingresa operacion fracciones (+, -, *, /)\\n");
    printf("a/b o c/b\\n");
    scanf(" %d %c %d %c %d %c %d" \
          , &a, &div_1, &b, &operation, &c, &div_2, &d);

    if(b != d)
    {
        printf("Ingresa un valor de denominadores igual\\n");
    }

    if(div_1 == '/' && div_2 == '/')
    {
        switch(operation)
        {
            case '+':
                resultado(a, b, operation, c, d, a+c, b);
                break;
            case '-':
                resultado(a, b, operation, c, d, a-c, b);
                break;
            case '*':
                resultado(a, b, operation, c, d, a*c, b * b);
                break;
            case '/':
                resultado(a, b, operation, c, d, a, c);
                break;
        }
    }
}
```

```

        default:
            printf("Ingresa un valor valido\n\n");
    }
}
else
{
    printf("Ingresa valores validos\n\n");
}

return 0;
}

```

Figure 8: resta

```

Ingresa operacion fracciones (+, -, *, /)
a/b o c/b
7/3-9/3
=====
7/3-9/3 = -2/3
=====

```

Figure 9: división

```

Ingresa operacion fracciones (+, -, *, /)
a/b o c/b
5/6/7/6
=====
5/6/7/6 = 5/7
=====

```

Figure 10: multiplicación

```

Ingresa operacion fracciones (+, -, *, /)
a/b o c/b
1/4*6/4
=====
1/4*6/4 = 6/16
=====

```

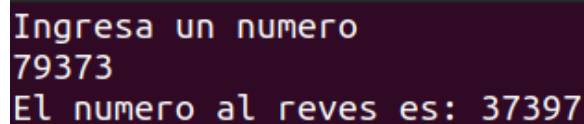
Ejercicio 4

Programa que imprima un número entero dado de n dígitos al revés. Ejem, entrada: 79373, salida: 37397.

Respuesta

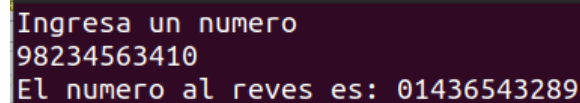
Se creó un programa que lee un string el cual debe ser un número, se realiza una comprobación para que esto sea así, de lo contrario indica que el dato de entrada es erróneo. Se utiliza una función recursiva para poder imprimir el número. En seguida se muestran algunos resultados y el código de la implementación.

Figure 11: Ejemplo 1



```
Ingresa un numero
79373
El numero al revés es: 37397
```

Figure 12: Ejemplo 2



```
Ingresa un numero
98234563410
El numero al revés es: 01436543289
```

Code 4: Problema 4

```
//Giovanny Encinia
// 23-08-2021
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define SIZE 20
#define ZERO 0

void voltea_numero(char *number, int place)
{
    /*funcion recursiva que imprime un numero
    al revés*/

    //mientras no termine el string
    if(number[place] != '\0')
    {
        //recorre el arreglo
        voltea_numero(number, place + 1);
        printf("%c", number[place]);
    }
}

int main(void)
{
    char number[SIZE];
    int i = ZERO;
```

```

printf("Ingresa un numero\n");
scanf(" %s", number);

//Comprueba que el valor ingresado es numero
while(number[i] != '\0')
{
    if(!isdigit(number[i]))
    {
        printf("El valor ingresado no es un numero\n");
        return ZERO;
    }

    i++;
}

printf("El numero al revés es: ");
voltea_numero(number, ZERO);
printf("\n");

return ZERO;
}

```

Ejercicio 5

Programa que evalúe la siguiente expresión:

$$e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-x^2)^n}{n!} = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} x^{2n}$$

Debe pedir el número de términos a evaluar.

Respuesta

Se creó un programa que utiliza memoization para poder realizar los cálculos, pienso que al aplicar esto el error debido a las constantes operaciones se disminuye. El programa en un inicio está habilitado para poder intentar hacer los cálculos con 100 términos, si se requiere realizar más términos, se debe cambiar el número de la declarativa *SIZE* a el número que se quiera, si se requieren 300 términos entonces *SIZE* debe tomar el valor de 600.

Como output el programa muestra el valor calculado con la función `exp()` de la librería estándar y el resultado de la serie de Maclaurin, además se muestra el error relativo existente entre ambos resultados.

Es importante mencionar que al ser una serie de Maclaurin los números alrededor de 0 serán los que necesiten menos términos de la serie para obtener un resultado con un valor relativo pequeño. Números alejados de 0 es posible que no converjan debido a que el exponente en x crece muy rápido y no se puede representar de manera correcta el número en la máquina.

En seguida se muestran algunos resultados obtenidos y el código del algoritmo en C.

Figure 13: Resultado 1

```
Ingrese el valor de x a evaluar
1
Ingrese el numero de iteraciones a realizar
15

El resultado real es: 0.3678794
El resultado de la serie es: 0.3678794
El error relativo es: 0.0000000000001
```

Figure 14: Resultado 2

```
Ingrese el valor de x a evaluar
-1
Ingrese el numero de iteraciones a realizar
13

El resultado real es: 0.3678794
El resultado de la serie es: 0.3678794
El error relativo es: 0.0000000000292
```

Figure 15: Resultado 3

```
Ingrese el valor de x a evaluar
2
Ingrese el numero de iteraciones a realizar
12

El resultado real es: 0.0183156
El resultado de la serie es: 0.0266688
El error relativo es: 0.4560673863791
```

Figure 16: resultado 4

```
Ingrese el valor de x a evaluar
1.4
Ingrese el numero de iteraciones a realizar
20

El resultado real es: 0.1408584
El resultado de la serie es: 0.1408584
El error relativo es: 0.0000000000002
```

Code 5: Problema 5

//giovanny Encinia
//23-08-2021


```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define SIZE 100// terms number of the mclaurin's serie
#define ONE 1
#define ZERO 0
#define E-R(e_x, result) (fabs(e_x - result)/fabs(e_x))

long factorial(long n, long *memo_fact);
double potencia(long n, double x, double *memo_pot);
double serie(long *memo_fact, double *memo_pot, double x, long n);

long factorial(long n, long *memo_fact)
{
    /*Calcula el factorial de manera recursiva
    Parametros
    =====
    n: termino del factorial
    memo_fact: es el arreglo donde se guardaran los resultados*/

    double result;

    if(memo_fact[n])
    {
        result = memo_fact[n];
    }
    else
    {
        result = n * factorial(n - ONE, memo_fact);
    }

    memo_fact[n] = result;

    return result;
}

double potencia(long n, double x, double *memo_pot)
{
    /*Calcula la potencia de manera recursiva
    Parametros
    =====
    n: termino del factorial
    x: es el numero que se estara elevando
    memo_pot: es el arreglo donde se guardaran los resultados

    */

    double result;

```

```

    if(memo_pot[n])
    {

        result = memo_pot[n];
    }
    else
    {
        result = potencia(n - ONE, x, memo_pot) * x;
    }

    memo_pot[n] = result;
    return result;
}

double serie(long *memo_fact, double *memo_pot, double x, long n)
{
    /*Calcula la serie, esta esta limitada a SIZE/2 terminos*/

    long i = 0;
    double result;

    while(i <= n)
    {
        if(i % 2 == 0)
        {
            result += potencia(2*i, x, memo_pot) / factorial(i, memo_fact);
        }
        else
        {
            result -= potencia(2*i, x, memo_pot) / factorial(i, memo_fact);
        }

        i++;
    }

    return result;
}

int main(void)
{
    long *memo_fact = (long *)calloc(SIZE, sizeof(long));
    double *memo_pot = (double *)calloc(SIZE, sizeof(double));

    double x, result, real;
    long n;

```

```

if(memo_fact == NULL || memo_pot == NULL)
{
    printf("Memoria llena");
    return ZERO;
}
else
{
    //inicializa los arreglos
    memo_fact[ONE] = memo_fact[ZERO] = ONE;
    memo_pot[ZERO] = ONE;

    printf("Ingrese el valor de x a evaluar\n");
    scanf("%lf", &x);

    real = exp(-pow(x, 2)); // calcula el valor "real"
    printf("Ingrese el numero de iteraciones a realizar\n");
    scanf("%ld", &n);
    printf("\n\n");
    result = serie(memo_fact, memo_pot, x, n);
    printf("El resultado real es: %.7f\n", real);
    printf("El resultado de la serie es: %.7f\n", result);
    printf("El error relativo es: %.13f\n", ER(real, result));
}

free(memo_fact);
free(memo_pot);

return ZERO;
}

```

Ejercicio 6

Programa que convierta un número decimal a cualquier base.

Respuesta

Se creó un algoritmo que como datos de entrada pide el numero a transformar y la base que se usará para transformarlo. El algoritmo imprime en pantalla el numero transformado. Cabe mencionar que el numero transformado tiene un máximo de 31 símbolos. Para bases mayores a 10, los símbolos que se imprimen son letras mayúsculas del abecedario, si la base supera a los 27, imprimirá símbolos de ANSI, los símbolos que puede imprimir son los que están en el ANSI.

En seguida se muestran algunos resultados y el código del programa.

Figure 17: Resultado 1

```
Ingresa el numero a transformar
16
Ingresa la base en la que se requiere transformar
17
El numero 16 en base 17 es:
5
```

Figure 18: Resultado 2

```
Ingresa el numero a transformar
78
Ingresa la base en la que se requiere transformar
2
El numero 78 en base 2 es:
1001110
```

Figure 19: Resultado 3

```
Ingresa el numero a transformar
6
Ingresa la base en la que se requiere transformar
3
El numero 6 en base 3 es:
20
```

Figure 20: Resultado 4

```
Ingresa el numero a transformar
43
Ingresa la base en la que se requiere transformar
16
El numero 43 en base 16 es:
2B
```

Code 6: Problema 6

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define ZERO 0
#define SIZE 31
#define CORRE 48
#define CORRE2 55

void conversion(int number, int base);
```

```

void conversion(int number, int base)
{

    int i = ZERO;
    int arreglo[SIZE];

    printf("El numero %d en base %d es:\n", number, base);

    while(number > ZERO)
    {

        arreglo[i] = number%base;
        number /= base;
        i++;
    }

    i--;

    while(i >= ZERO)
    {
        if(arreglo[i] > 9)
            printf("%c", arreglo[i] + CORRE_2);
        else
            printf("%c", arreglo[i] + CORRE);
        i--;
    }
}

int main(void)
{
    int i = ZERO, base, number;
    int *result;

    printf("Ingresa el numero a transformar\n");
    scanf("%d", &number);
    printf("Ingresa la base en la que se requiere transformar\n");
    scanf("%d", &base);
    printf("\n");
    conversion(number, base);
    printf("\n");

    return ZERO;
}

```

Ejercicio 7

Programa que acepte un fracción del tipo a/b tal que $(a, b) \in \mathbb{N}$, y la reduzca a sus términos mínimos. Ejem: $\frac{6}{24} \rightarrow \frac{1}{4}$. Debe encontrar el GCD.

Respuesta

Los datos de entrada del algoritmo se leen con un `scanf()` el cual almacena un entero, un char y un entero, en ese orden, una vez hecho esto, se utiliza el algoritmo de Euclides para encontrar el GCD, de una manera recursiva, una vez realizado esto el GCD, se resta al denominador y numerador del dato de entrada y se regresa el resultado a la pantalla.

en seguida se muestran algunos resultados y el codigo del problema.

Figure 21: Resultado 1

```
Ingrese una fraccion de la forma a/b
3/39
=====
3/39 = 1/13
=====
```

Figure 22: Resultado 2

```
Ingrese una fraccion de la forma a/b
2/1024
=====
2/1024 = 1/512
=====
```

Figure 23: Resultado 3

```
Ingrese una fraccion de la forma a/b
512/1024
=====
512/1024 = 1/2
=====
```

Figure 24: Resultado 4

```
Ingrese una fraccion de la forma a/b
6/27
=====
6/27 = 2/9
=====
```

Figure 25: Resultado 5

```
Ingrese una fraccion de la forma a/b
-27/81
=====
-27/81 = 1/-3
=====
```

Code 7: Problema 7

```
//Giovanny Encinia
//24-08-2021
#include <stdio.h>
#include <stdlib.h>
#define ZERO 0

int mcd(int a, int b);

int mcd(int a, int b)
{
    /*Encuentra el maximo comun divisor de dos numeros*/
    int temp;

    if(b == ZERO)
    {
        return a;
    }

    else
    {
        if(b > a)
        {
            temp = a;
            a = b;
            b = temp;
        }
        return mcd(b, a % b);
    }
}

int main(void)
{
    int a, b, den;
    char div;

    printf("Ingrese una fraccion de la forma a/b\n");
    scanf(" %d %c %d", &a, &div, &b);

    if(div != '/')
    {
        printf("Ingrese una valor valido\n");
        printf("Fraccion en la forma a/b\n");
    }

    return 0;
}

den = mcd(a, b);
printf("=====\\n");
```

```
printf("          %d/%d = %d/%d\n",  a, b, a/den, b/den);  
printf("=====\\n");  
  
return 0;  
}
```
