Práctica 6

Argumentos al *main* – Preprocesador – Bibliotecas

Argumentos al *main*

Nota: en el tutorial de Code::Blocks se explica cómo pasar argumentos a la función main en dicho entorno.

1. Analice, compile y ejecute el siguiente código:

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

int main (int argc, char * argv[]) {
      printf("\nargc = %d",argc);
      printf("\nargv[0] => %s",argv[0]);
      return 0;
}
```

   ¿Qué imprime? ¿Por qué?

2. Escriba un programa que calcule e imprima el promedio de 4 números que se pasan como argumentos a la función *main*. En caso de no pasar la cantidad necesaria de parámetros, informar al usuario.

3. a. Investigue las funciones de conversión de texto a número y escriba un programa "calculadora.c" que reciba dos números enteros y un operador como argumentos a la función main e imprima en pantalla el resultado de la operación. El operador debe ser: "+", "-", "/", ".". Si el operador recibido no es uno de los mencionados anteriormente, entonces el programa debe imprimir un mensaje de error.

   b. Utilice una terminal para compilar el programa a través de la línea de comandos.

Preprocesador

4. Analice y ejecute el siguiente código:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define SQUAREOF(x) x*x

int main() {
    int xin=3;

    printf("\nxin=%i",xin);
    printf("\nSQUAREOF(xin)=%i", SQUAREOF(xin));
    printf("\nSQUAREOF(xin+4)=%i", SQUAREOF(xin+4));
    printf("\nSQUAREOF(xin+xin)=%i", SQUAREOF(xin+xin));
}
```

   a. ¿Qué es lo que hace?
   b. La macro `SQUAREOF` debería calcular el cuadrado de un número. ¿Los resultados son los esperados? ¿Cuál es el problema?
   c. Modifique la macro para que el cálculo sea correcto.

5. Analice y ejecute el siguiente código:

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

#define size 25

void imprimir(int * v, int size){
    int i;
    for (i=0; i<size; i++){
        printf("v[%d]= %d", i, v[i]);
    }
}

int main(){
    int v[size];
    imprimir(v, size);
    return 0;
}
```

a) ¿El código compila? Lea detenidamente el primer error que se detecta. ¿Cómo se explica?
b) Identifique las etapas de *precompilación* y *compilación*. ¿Qué es lo que está ocurriendo en cada una?
c) ¿Cómo podría solucionarse este problema?

6. Realice un programa que defina una macro `AREA_CIRCULO(r)` que permita calcular el área de un círculo para un radio *r*. El cuerpo del programa debe generar aleatoriamente 10 valores diferentes que representan radios e imprimir el área correspondiente.

7. El siguiente código debe imprimir el máximo entre las variables a y b. Complete la definición de la macro para lograr el objetivo.

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX(x,y) _____________________

int main () {
    int a, b;

    scanf("%d%d", &a, &b);
    printf("\n MAX(%d, %d) = %d", a, b, MAX(a,b));

    return 0;
}
```

7. Analice y ejecute el siguiente código:

```c
#include <stdio.h>
#include <stdlib.h>
#define DEBUG 0

int main(){
    int x= 15;

    while (x<50){
        #if DEBUG
            printf("x= %d", x);
            y= y+1;
        #endif
        x++;
    }
    return 0;
}
```

a) ¿El código compila? ¿Por qué?
b) ¿Qué sucedería al cambiar la constante simbólica `DEBUG` al valor 1? Evalúe el comportamiento del código. ¿Qué beneficios trae en este caso la compilación condicional?
c) ¿Es posible cambiar el valor de la constante simbólica `DEBUG` en tiempo de ejecución?
d) ¿Qué sucede si en lugar de utilizar la directiva `#if` se emplea `#ifdef`? ¿Incide el valor de la constante simbólica `DEBUG` en el resultado de la compilación?

8. Realice un programa que genere un vector de 1000 enteros al azar y utilice una función para localizar un elemento. Se pide:
   a) Implementar la función de búsqueda que, dado un entero, retorne la posición donde se encuentra. En caso de no existir el número retorne -1.
   b) Modificar a) para agregar información de depuración que permita imprimir en consola la cantidad de veces que se debió acceder al arreglo para encontrar (o no) el elemento en cada llamado a la función. Utilice las directivas del procesador para activar/desactivar la depuración e imprimir/no imprimir la información en la consola.
   c) Verifique el tamaño del programa compilando con la depuración y sin la depuración. Explique porque difieren.

Bibliotecas

9. Implemente la biblioteca `imath.h`, la cual contiene funciones matemáticas para enteros. La misma debe contener las siguientes funciones:

   a. `int potencia(int x, int y) – Eleva x a la y-ésima potencia.`
   b. `int cuadrado(int x) – Eleva x al cuadrado.`
   c. `int cubo(int x) – Eleva x al cubo.`
   d. `int absoluto(int i) – Retorna el valor absoluto entero de i.`
   e. `int factorial (int i) – Retorna el factorial de i.`
   f. `int sumatoria (int i) – Retorna la sumatoria de 0 hasta i.`
   g. `int multiplo (int x, int y) – Retorna 1 si y es divisor de x, 0 en otro caso.`
   h. `int divisor (int x, int y) – Retorna 1 si y es múltiplo de x, 0 en otro caso.`
   i. `int par (int i) – Retorna 1 si i es par, 0 en otro caso.`
   j. `int impar (int i) – Retorna 1 si i es impar, 0 en otro caso.`

   Una vez implementadas todas las funciones de biblioteca, escriba un programa que lea números enteros e imprima el cuadrado y el cubo de aquellos números que son pares y el factorial de aquellos que son impares. La lectura finaliza con el ingreso del 0.

10. a. Implemente la biblioteca `istack.h`, la cual debe contener tipos y operaciones necesarias para manipular una estructura de datos pila que almacene números enteros. Una pila es una estructura de datos homogénea y dinámica. El acceso a la misma se dice que es de tipo LIFO (Last In First Out), lo que quiere decir que los elementos se recuperan en el orden inverso en que fueron insertados. Implemente al menos las siguientes funciones:

    a. `Stack* s_create () – Retorna una nueva pila. Se debe invocar antes de manipular cualquiera de ellas.`
    b. `int s_push(stack* s, int n) – Apila n en s. Retorna el elemento apilado.`
    c. `int s_pop (stack* s) – Desapila un elemento de s.`
    d. `int s_top (stack s) – Retorna el próximo elemento que será desapilado.`
    e. `int s_empty(stack s) – Retorna 1 si s está vacía, 0 en caso contrario.`
    f. `int s_length(stack s) – Retorna la cantidad de elementos apilados en s.`

b. Una vez implementadas todas las funciones de biblioteca, escriba un programa que lea números enteros y los inserte en una pila. A continuación, desapile los elementos para verificar si funciona correctamente. La lectura finaliza con el ingreso del 0.

c. Manteniendo la misma interfaz, modifique el tipo de datos ***stack*** y reimplemente la biblioteca para que la función ***s_length*** retorne la cantidad de elementos sin recorrer la pila.

11. Implemente una biblioteca `diccionario.h` que permita manejar un diccionario de palabras. Determine las estructuras de datos necesarias para implementar la biblioteca. Las operaciones que debe implementar son las siguientes:
    a) Crear un diccionario: inicializa la estructura del diccionario.
    b) Agregar palabra: agrega una nueva palabra al diccionario siempre y cuando no exista. Retorna si la agregó o no.
    c) Existe palabra: determina si una palabra está en el diccionario o no.
    d) Eliminar palabra: elimina una palabra del diccionario. Retorna si la palabra fue eliminada o no.
    e) Cargar desde un archivo: carga un diccionario desde un archivo de texto.
    f) Guardar a un archivo: guarda un diccionario en un archivo de texto.
    g) Destruir el diccionario: libera los recursos del diccionario.

    Una vez implementadas todas las funciones de biblioteca, escriba un programa que permita probarlas.