Introducción a la Programación Practica 7

Medina Martinez Jonathan Jason 2023640061

20 de mayo del 2023

Índice

1. Objetivo

Desarrollo de programas utilizando los apuntadores para el paso de parámetros por valor y por referencia a funciones.

2. Introducción

La programación es una herramienta fundamental en el desarrollo de software, y el dominio de conceptos avanzados como el uso de apuntadores resulta esencial para optimizar el rendimiento y la eficiencia de los programas. En esta práctica, nos enfocaremos en el uso de apuntadores para el paso de parámetros por valor y por referencia a funciones. El objetivo principal es desarrollar programas que utilicen apuntadores de manera efectiva para realizar operaciones como la suma de números, la manipulación de arreglos y el procesamiento de cadenas de caracteres. A lo largo de esta práctica, exploraremos diversos ejercicios que nos permitirán familiarizarnos con el manejo adecuado de apuntadores.

3. Desarrollo

3.1. Problema 1

Programa que solicite al usuario dos números y devuelva la suma de los mismos, utilizando punteros.

3.1.1. Programa1.c

```c
/**
* @file Programa1.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-19
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

void sumarNumeros(int num1, int num2, int *resultado);

int main() {
	int num1, num2, resultado;

	printf("Ingrese el primer numero: ");
	scanf("%d", &num1);
	printf("Ingrese el segundo numero: ");
	scanf("%d", &num2);

	sumarNumeros(num1, num2, &resultado);

	printf("La suma es: %d\n", resultado);

	return 0;
}

/// @brief Funcion que suma dos numeros utulizando punteros.
/// @param num1 El primer numero.
/// @param num2 El segundo numero.
/// @param resultado El resultado.
void sumarNumeros(int num1, int num2, int *resultado) {
	*resultado = num1 + num2;
}
```

3.1.2. Ejecución

```
Ingrese el primer numero: 5
Ingrese el segundo numero: 20
La suma es: 25
```

```
Ingrese el primer numero: 5
Ingrese el segundo numero: 16
La suma es: 21
```

```
Ingrese el primer numero: 10
Ingrese el segundo numero: 18
La suma es: 28
```

```
Ingrese el primer numero: 12
Ingrese el segundo numero: 96
La suma es: 108
```

```
Ingrese el primer numero: 175
Ingrese el segundo numero: 236
La suma es: 411
```

```
Ingrese el primer numero: 892
Ingrese el segundo numero: 568
La suma es: 1460
```

3.2. Problema 2

Programa que solicite al usuario dos números y devuelva la suma de los mismos, utilizando paso de parámetros por referencia.

3.2.1. Programa2.c

```c
/**
* @file Programa2.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-19
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

void sumarNumeros(int *num1, int *num2, int *resultado);

int main() {
	int num1, num2, resultado;

	printf("Ingrese el primer numero: ");
	scanf("%d", &num1);

	printf("Ingrese el segundo numero: ");
	scanf("%d", &num2);

	sumarNumeros(&num1, &num2, &resultado);

	printf("La suma es: %d\n", resultado);

	return 0;
}

/// @brief Funcion que suma dos numeros utulizando punteros.
/// @param num1 El primer numero.
/// @param num2 El segundo numero.
/// @param resultado El resultado.
void sumarNumeros(int *num1, int *num2, int *resultado) {
	*resultado = *num1 + *num2;
}
```

3.2.2. Ejecución

```
Ingrese el primer numero: 10
Ingrese el segundo numero: 15
La suma es: 25
```

```
Ingrese el primer numero: 23
Ingrese el segundo numero: 65
La suma es: 88
```

```
Ingrese el primer numero: 56
Ingrese el segundo numero: 87
La suma es: 143
```

```
Ingrese el primer numero: 1236
Ingrese el segundo numero: 1458
La suma es: 2694
```

```
Ingrese el primer numero: 789
Ingrese el segundo numero: 254
La suma es: 1043
```

```
Ingrese el primer numero: 457
Ingrese el segundo numero: 159
La suma es: 616
```

3.3. Problema 3

Programa que solicite al usuario n cantidad de números que sean almacenados en un arreglo e imprima el contenido del arreglo utilizando un puntero.

3.3.1. Programa3.c

```c

/**
* @file Programa3.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-20
*
* @copyrigth GPlv3
*
*/

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

int main() {
	int n;

	printf("Ingrese la cantidad de numeros a almacenar: ");
	scanf("%d", &n);

	int *arreglo = (int *)malloc(n * sizeof(int));

	if (arreglo == NULL) {

		printf("Error al asignar memoria.");

		return 1;

	}

	printf("Ingrese los numeros:\n");

	for (int i = 0; i < n; i++) {

		scanf("%d", arreglo + i);

	}

```

```
        printf("Contenido del arreglo:\n");

        for (int i = 0; i < n; i++) {

                printf("%d ", *(arreglo + i));

        }

        free(arreglo);

        return 0;
}
```

3.3.2. Ejecución

```
        Ingrese la cantidad de numeros a almacenar: 10
        Ingrese los numeros:
        1
        1
        156
        21
        242
        54
        57
        64
        57
        54
        Contenido del arreglo:
        1 1 156 21 242 54 57 64 57 54
```

```
        Ingrese la cantidad de numeros a almacenar: 100000000000000000000
        Error al asignar memoria.
```

```
        Ingrese la cantidad de numeros a almacenar: 7
        Ingrese los numeros:
        15
        25
        78
        98
        63
        32
        45
        Contenido del arreglo:
        15 25 78 98 63 32 45
```

3.4. Problema 4

Usando memoria dinámica, escriba un programa que solicite n números al usuario y devuelva el número más grande.

3.4.1. Programa4.c

```c

/**
* @file Programa4.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-20
*
* @copyrigth GPlv3
*
*/

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

int main() {
	int n;

	printf("Ingrese la cantidad de numeros: ");
	scanf("%d", &n);

	int *numeros = (int *)malloc(n * sizeof(int));

	if (numeros == NULL) {

		printf("Error al asignar memoria.");

		return 1;

	}

	printf("Ingrese los numeros:\n");

	for (int i = 0; i < n; i++) {

		scanf("%d", numeros + i);

	}

```

```
        int maximo = *numeros;

        for (int i = 1; i < n; i++) {

                if (*(numeros + i) > maximo) {

                        maximo = *(numeros + i);

                }

        }

        printf("El numero mas grande es: %d\n", maximo);

        free(numeros);

        return 0;
}
```

3.4.2. Ejecución

```
        Ingrese la cantidad de numeros: 8
        Ingrese los numeros:
        12
        32
        25
        23
        26
        52
        36
        15
        El numero mas grande es: 52
```

```
        Ingrese la cantidad de numeros: 5
        Ingrese los numeros:
        15
        995
        559
        1175
        45
        El numero mas grande es: 1175
```

```
        Ingrese la cantidad de numeros: 1000000000000000000
        Error al asignar memoria.
```

3.5. Problema 5

Programa que solicite al usuario una cadena y cuente el total de vocales y consonantes utilizando punteros.

3.5.1. Programa5.c

```c
/**
* @file Programa5.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-20
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

int contarVocalesConsonantes(char *cadena, int *vocales, int *consonantes);

int main() {
	char cadena[100];
	int vocales, consonantes, total;

	printf("Ingrese una cadena: ");
	fgets(cadena, sizeof(cadena), stdin);

	total = contarVocalesConsonantes(cadena, &vocales, &consonantes);

	printf("Cantidad de vocales: %d\n", vocales);
	printf("Cantidad de consonantes: %d\n", consonantes);
	printf("Total de caracteres: %d\n", total);

	return 0;
}

/// @brief Funcion que cuenta el numero de vocales y consonantes en una cadena de caracteres.
/// @param cadena La cadena de caracteres.
/// @param vocales Un puntero para almacenar el conteo de vocales.
/// @param consonantes Un puntero para almacenar el conteo de consonantes.
/// @return El total de caracteres (suma de vocales y consonantes).
int contarVocalesConsonantes(char *cadena, int *vocales, int *consonantes) {
	*vocales = 0;
```

```
39          *consonantes = 0;
40
41          while (*cadena != '\0') {
42                  if (*cadena == 'a' || *cadena == 'e' || *cadena == 'i' || *cadena == 'o
                        ' || *cadena == 'u' ||
43                  *cadena == 'A' || *cadena == 'E' || *cadena == 'I' || *cadena == 'O' ||
                        *cadena == 'U') {
44                          (*vocales)++;
45                  } else if ((*cadena >= 'a' && *cadena <= 'z') || (*cadena >= 'A' && *
                        cadena <= 'Z')) {
46                          (*consonantes)++;
47                  }
48                  cadena++;
49          }
50
51          return (*vocales + *consonantes);
52  }
```

3.5.2. Ejecución

```
1           Ingrese una cadena: Hola Mundo
2           Cantidad de vocales: 4
3           Cantidad de consonantes: 5
4           Total de caracteres: 9
```

```
1           Ingrese una cadena: Jonathan Jason Medina Martinez
2           Cantidad de vocales: 11
3           Cantidad de consonantes: 16
4           Total de caracteres: 27
```

```
1           Ingrese una cadena: ASDkfkdsmfkssdfaJAFKJJAfdfks
2           Cantidad de vocales: 4
3           Cantidad de consonantes: 24
4           Total de caracteres: 28
```

```
1           Ingrese una cadena: huask asjfjadknf asASnJSf akasfa
2           Cantidad de vocales: 9
3           Cantidad de consonantes: 20
4           Total de caracteres: 29
```

```
1           Ingrese una cadena: asfmlamfaasfal lasflma
2           Cantidad de vocales: 7
3           Cantidad de consonantes: 14
4           Total de caracteres: 21
```

3.6. Problema 6

Programa que genere un arreglo de n números pseudoaleatorios entre -1000 y 1000, donde n será proporcionado por el usuario. Enseguida, proceda a ordenar el arreglo de menor a mayor. Deberá utilizar memoria dinámica y punteros.

3.6.1. Programa6.c

```c
/**
* @file Programa6.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-20
*
* @copyrigth GPlv3
*
*/

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

void generarNumerosAleatorios(int *arreglo, int n);

void ordenarArreglo(int *arreglo, int n);

void imprimirArreglo(int *arreglo, int n);

int main() {
	int n;
	int *arreglo;

	printf("Ingrese la cantidad de numeros a generar: ");
	scanf("%d", &n);

	arreglo = (int *)malloc(n * sizeof(int));

	if (arreglo == NULL) {
		printf("Error al asignar memoria.");
		return 1;
	}

	generarNumerosAleatorios(arreglo, n);

	printf("Arreglo generado:\n");
```

```
39              imprimirArreglo(arreglo, n);
40
41              ordenarArreglo(arreglo, n);
42
43              printf("Arreglo ordenado:\n");
44              imprimirArreglo(arreglo, n);
45
46              free(arreglo);
47
48              return 0;
49      }
50
51      /// @brief Genera numeros pseudoaleatorios y los almacena en un arreglo.
52      /// @param arreglo El arreglo donde se almacenaran los numeros generados.
53      /// @param n La cantidad de numeros a generar.
54      void generarNumerosAleatorios(int *arreglo, int n) {
55              srand(time(NULL));
56
57              for (int i = 0; i < n; i++) {
58                      arreglo[i] = rand() % 2001 - 1000;
59              }
60      }
61
62      /// @brief Ordena un arreglo de menor a mayor.
63      /// @param arreglo El arreglo a ordenar.
64      /// @param n La longitud del arreglo.
65      void ordenarArreglo(int *arreglo, int n) {
66              int *p, *q, temp;
67
68              for (p = arreglo; p < arreglo + n - 1; p++) {
69                      for (q = p + 1; q < arreglo + n; q++) {
70                              if (*p > *q) {
71                                      temp = *p;
72                                      *p = *q;
73                                      *q = temp;
74                              }
75                      }
76              }
77      }
78
79      /// @brief Imprime los elementos de un arreglo.
80      /// @param arreglo El arreglo a imprimir.
81      /// @param n La longitud del arreglo.
82      void imprimirArreglo(int *arreglo, int n) {
```

```
            for (int i = 0; i < n; i++) {
                    printf("%d ", arreglo[i]);
            }
            printf("\n");
    }
```

3.6.2. Ejecución

```
    Ingrese la cantidad de numeros a generar: 10
    Arreglo generado:
    -989 784 85 -231 -502 248 -267 309 137 384
    Arreglo ordenado:
    -989 -502 -267 -231 85 137 248 309 384 784
```

```
    Ingrese la cantidad de numeros a generar: 15
    Arreglo generado:
    -967 482 65 -382 -591 970 -491 580 654 120 359 478 1000 -724 -887
    Arreglo ordenado:
    -967 -887 -724 -591 -491 -382 65 120 359 478 482 580 654 970 1000
```

```
    Ingrese la cantidad de numeros a generar: 4
    Arreglo generado:
    -333 527 310 838
    Arreglo ordenado:
    -333 310 527 838
```

```
    Ingrese la cantidad de numeros a generar: 12
    Arreglo generado:
    -255 -902 -935 540 -248 -974 -100 -967 -297 -673 883 75
    Arreglo ordenado:
    -974 -967 -935 -902 -673 -297 -255 -248 -100 75 540 883
```

```
    Ingrese la cantidad de numeros a generar: 7
    Arreglo generado:
    -127 306 -365 -506 49 334 219
    Arreglo ordenado:
    -506 -365 -127 49 219 306 334
```

```
    Ingrese la cantidad de numeros a generar: 9
    Arreglo generado:
    -13 -708 288 -253 611 695 845 -215 -860
    Arreglo ordenado:
    -860 -708 -253 -215 -13 288 611 695 845
```

3.7. Problema 7

Programa que solicite al usuario n números que serán almacenados en un arreglo, y devuelva la suma de todos los elementos del arreglo, utilizando punteros.

3.7.1. Programa7.c

```c
/**
* @file Programa7.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-20
*
* @copyrigth GPlv3
*
*/

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

int calcularSuma(int *arreglo, int n);

int main() {
	int n;

	printf("Ingrese la cantidad de numeros: ");
	scanf("%d", &n);

	int *arreglo = (int *)malloc(n * sizeof(int));

	if (arreglo == NULL) {
		printf("Error al asignar memoria.\n");
		return 1;
	}

	printf("Ingrese los numeros:\n");
	for (int i = 0; i < n; i++) {
		scanf("%d", &arreglo[i]);
	}

	int suma = calcularSuma(arreglo, n);

	printf("La suma de los numeros es: %d\n", suma);

	free(arreglo);
```

```
40
41          return 0;
42  }
43
44  /// @brief Calcula la suma de los elementos de un arreglo.
45  /// @param arreglo El arreglo de numeros.
46  /// @param n La longitud del arreglo.
47  /// @return La suma de los elementos del arreglo.
48  int calcularSuma(int *arreglo, int n) {
49          int suma = 0;
50          int *p = arreglo;
51
52          for (int i = 0; i < n; i++) {
53                  suma += *p;
54                  p++;
55          }
56
57          return suma;
58  }
```

3.7.2. Ejecución

```
1          Ingrese la cantidad de numeros: 10000000000000000
2          Error al asignar memoria.
```

```
1          Ingrese la cantidad de numeros: 8
2          Ingrese los numeros:
3          15
4          151
5          574
6          85
7          586
8          75457
9          5575
10         57
11         La suma de los numeros es: 82500
```

```
1          Ingrese la cantidad de numeros: 4
2          Ingrese los numeros:
3          155882
4          282254845
5          629
6          2585956
7          La suma de los numeros es: 284997312
```

3.8. Problema 8

Programa que permita al usuario escribir un texto del tamaño que desee, que será almacenado en un arreglo de caracteres. Una vez que el usuario presione punto (.), el programa deberá imprimir el contenido del arreglo y terminar. Para esto, deberá utilizar memoria dinámica.

3.8.1. Programa8.c

```c
/**
* @file Programa8.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-05-21
*
* @copyrigth GPlv3
*
*/

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

#define MAX_LENGTH 1000

int main() {
	char *texto = (char *)malloc(MAX_LENGTH * sizeof(char));

	if (texto == NULL) {
		printf("Error al asignar memoria.\n");
		return 1;
	}

	printf("Escriba un texto (presione punto para finalizar):\n");

	char *p = texto;
	char c;

	while ((c = getchar()) != '.') {
		*p = c;
		p++;

		if (p - texto >= MAX_LENGTH - 1) {
			printf("Tamanio maximo alcanzado. Se truncara el texto.\n");
			break;
		}
```

```
        }

        *p = '\0';

        printf("Texto ingresado:\n%s\n", texto);

        free(texto);

        return 0;
}
```

3.8.2. Ejecución

```
        Escriba un texto (presione punto para finalizar):
        Hola mundo.
        Texto ingresado:
        Hola mundo
```

```
        Escriba un texto (presione punto para finalizar):
        juan lopez gonzales. hola
        Texto ingresado:
        juan lopez gonzales
```

```
        Escriba un texto (presione punto para finalizar):
        gnfsnndkngksdkgs gsdkkmsdgs dgdsgjdskgjds gsdjgs. sdjfigndsjjdng gsgdsg
        Texto ingresado:
        gnfsnndkngksdkgs gsdkkmsdgs dgdsgjdskgjds gsdjgs
```

```
        Escriba un texto (presione punto para finalizar):
        kdafdskmflks dfksnkg dmkfmksdgls dsomgsdkg sdgds.gsdgksdngkmsdg
        Texto ingresado:
        kdafdskmflks dfksnkg dmkfmksdgls dsomgsdkg sdgds
```

4. Conclusión

En conclusión, en esta práctica hemos adquirido experiencia en el uso de apuntadores en el desarrollo de programas. Hemos explorado la utilidad de los apuntadores al realizar operaciones como la suma de números, la manipulación de arreglos, la búsqueda del número más grande, el procesamiento de cadenas de caracteres, entre otros. A través de ejercicios prácticos, hemos comprendido cómo los apuntadores nos permiten acceder y modificar directamente la memoria de un programa, lo que resulta en un mejor rendimiento y eficiencia en la ejecución de tareas.