	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Adrian Ulises Mercado Martinez
Asignatura:	Fundamentos de programación.
Grupo:	10
No de Práctica(s):	12
Integrante(s):	González Bretón Arturo Paniagua Bautista Daniel Castañeda Mora Carlos
No. de Equipo de cómputo empleado:	49-50-51
No. de Lista o Brigada:	14
Semestre:	1
Fecha de entrega:	19/11/19
Observaciones:	

CALIFICACIÓN: _____

Práctica12 Funciones.

Castañeda Mora Carlos
González Bretón Arturo
Paniagua Bautista Daniel

17 octubre 2019

1 Introduction

La modularización, es una técnica usada por los programadores para hacer sus códigos más cortos, ya que consiste en reducir un gran problema complejo, en pequeños problemitas más sencillos, concentrándose en la solución por separado, de cada uno de ellos.

En C, se conocen como funciones aquellos trozos de códigos utilizados para dividir un programa con el objetivo que, cada bloque realice una tarea determinada.

En las funciones juegan un papel muy importe las variables, ya que como se ha dicho estas pueden ser locales o globales.

Variables Globales: Estas se crean durante toda la ejecución del programa, y son globales, ya que pueden ser llamadas, leídas, modificadas, etc; desde cualquier función. Se definen antes del main().

Variables Locales: Estas, pueden ser utilizadas únicamente en la función que hayan sido declaradas. A menudo, se utiliza el adjetivo de "Subprogramas", para referirse a las funciones. Los subprogramas se comunican con el programa principal, que es el que contiene a las funciones, mediante parámetros, que estos pueden ser: Parámetros Formales y Parámetros Actuales.

Cuando se da la comunicación los parámetros actuales son utilizados en lugar de los parámetros formales.

2 Desarrollo

1)Primero se crea una estructura de tipo complex para guardar los números complejos.

```c
#ifndef COMPLEX_H
#define COMPLEX_H

#include<stdio.h>

struct _complex{
        float real;
        float img;
};

typedef struct _complex complex;



#endif
```

2)Se crea un .h que almacena todas las funciones que se van a usar, además de otros .h que uilizará el código.

```h
#ifndef FUNCIONES_H
#define FUNCIONES_H

#include<stdio.h>
#include "complex.h"

complex suma(complex a, complex b);
complex resta(complex a, complex b);
complex multi(complex a, complex b);
complex div(complex a, complex b);
complex conj(complex a);
void leer_complex(complex *a);



int menu();
void print_complex(complex a);

#endif
```

3)En este .c se agrega el código ara cada función.

```c
#include "funciones.h"


complex suma(complex a,complex b){
        complex r;
        r.real=a.real+b.real;
        r.img=a.img+b.img;
        return r;
}

complex resta(complex a, complex b){
        complex r;
        r.real=a.real-b.real;
        r.img=a.img-b.img;
        return r;
}
complex multi(complex a, complex b){
        complex r;
        r.real=(a.real*b.real)-(a.img*b.img);
        r.img=(a.real*b.img)+(a.img*b.real);
        return r;
}

complex div(complex a,complex b){
        complex r;
        r.real=((a.real*b.real)+(a.img*b.img))/(b.real*b.real+b.img+b.img);
        r.img=((a.img*b.real)-(a.real*b.img))/(b.real*b.real+b.img+b.img);
        return r;
}
complex conj(complex a){
        complex r;
        r.real=a.real;
        r.img=-a.img;
        return r;
}
```

4)Al programar la función principal sólo hay que incluir el .h de nuestras funciones, pues en el referimos todas las bibliotecas que vamos a usar. Además se crea la estructura del programa, el cual será ejecutada y llamara las funciones que creamos.

```c
#include "funciones.h"

int main(){
        complex a,b;
        complex res;
        int x;
        x=0;
        while(x!=6){
                x=menu();
                switch(x){
                        case 1:
                                leer_complex(&a);
                                leer_complex(&b);
                                res= suma(a,b);
                                print_complex(res);
                                break;
                        case 2:
                                leer_complex(&a);
                                leer_complex(&b);
                                res= resta(a,b);
                                print_complex(res);
                                break;
                        case 3:
                                leer_complex(&a);
                                leer_complex(&b);
                                res= multi(a,b);
                                print_complex(res);
                                break;
                        case 4:
                                leer_complex(&a);
                                leer_complex(&b);
                                res= div(a,b);
                                print_complex(res);
                                break;
                        case 5:
                                leer_complex(&a);

                                res= conj(a);
```

3 Conclusión

Las funciones son sumamente útiles para facilitar el entendimiento del código y por lo tanto, su actualización, corrección y mantenimiento; sin embargo hay que considerar también que mientras más funciones tenga el programa, más se entorpecerá al convertirse el proceso, por lo que hay que utilizarlas de una forma eficiente. Los objetivos de esta práctica se cumplieron en su totalidad, los programas creados fueron divididos en funciones y en archivos diferentes; se identificaron adecuadamente los parámetros y los prototipos de las funciones, así como el cuerpo de cada una de ellas. Arturo

Aprendí que las funciones nos ayudan a reciclar el código y también para un mejor entendimiento del mismo, la sintaxis básica para definir una función es el valor de retorno, pronombre para identificar la función, los argumentos de entrada y después de esto sigue el bloque de instrucciones. La función principal llamada main también puede recibir parámetros y para utilizarlos es necesario indicar los parámetros al momento de ejecutar un programa en la terminal. Daniel

Dividir un programa en funciones es muy imprtante, pues es más fácil que un equipo de personas colabore con ese proyecto, es más fácil encontrar algún error en una función, si esa función no funciona, es más fácil que alguien te ayude a identificar ese error en una función que en todo el main, además de que da elegancia al código que programas. Pues un buen programador tiene que saber crear el código en varios archivos, para que la edición de estos sea más fácil y dnamica en el futuro, saber que parte del programa contiene a que cosa. Carlos.

4 Bibliografía

http://programandoenc.over-blog.es/article-32481588.html https://es.wikibooks.org/wiki/Programaci