

Implementación de la estructura de control lógica selectiva con funciones

Estudiante:   
Cristian Miranda  
Módulo o Unidad de Aprendizaje:  
Programación de Sistemas informáticos  
Fecha: 04-06-23  
otro: Otros

Actividad 5

## **Introducción**

¡Bienvenido al mundo de la resolución de problemas informáticos! En este documento, te ofrezco la oportunidad de adentrarte en el fascinante campo de la construcción de algoritmos computacionales. ¿Qué te parecería utilizar pseudocódigos que combinan lógica matemática con los componentes de software necesarios para resolver problemas prácticos?

Nuestro enfoque se basa en aplicar buenas prácticas de codificación para asegurar la eficiencia y robustez de tus algoritmos. Podrás analizar, diseñar e implementar soluciones para una amplia variedad de aplicaciones, donde podrás determinar aspectos específicos a calcular y explorar diferentes figuras geométricas.

Además, te invito a expandir tus habilidades investigativas, ya que en ocasiones deberás buscar y descubrir datos relacionados con las figuras que no se mencionan explícitamente en el enunciado del problema. De esta manera, podrás abordar desafíos más complejos y obtener soluciones aún más completas.

¡Únete y desata tu creatividad en el mundo de los algoritmos computacionales!

Tabla de contenido

[**Introducción** 1](#_Toc136823061)

[**Tabla de Figuras** 2](#_Toc136823062)

[**Implementación de la estructura de control lógica secuencial** 4](#_Toc136823063)

[13. Ortoedro 4](#_Toc136823064)

[14. Cono 6](#_Toc136823065)

[15. Prisma 8](#_Toc136823066)

[**Conclusión** 10](#_Toc136823067)

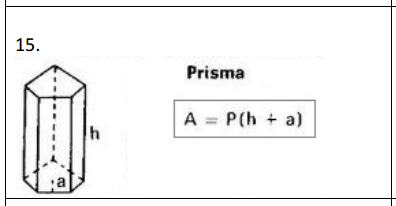
## **Tabla de Figuras**

Imagen que contiene Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza baja



**Desarrollo**

Las figuras geométricas son elementos fundamentales en el estudio de la geometría, una rama de las matemáticas que se dedica al análisis y la descripción de las propiedades y relaciones espaciales de los objetos. Estas figuras se encuentran presentes en nuestro entorno cotidiano y son de vital importancia en diversos campos, como la arquitectura, el diseño, la ingeniería y la física.

En geometría, las figuras geométricas se clasifican en dos dimensiones (2D) y tres dimensiones (3D). Las figuras 2D son aquellas que tienen solo largo y ancho, mientras que las figuras 3D tienen largo, ancho y altura, lo que les confiere una mayor complejidad y profundidad.

Entre las figuras geométricas más comunes en el plano 2D se encuentran el círculo, el triángulo, el cuadrado, el rectángulo, el rombo, el trapecio, el pentágono, el hexágono y el octógono, entre otros. Cada una de estas figuras tiene características específicas que las distinguen, como la longitud de sus lados, el número de ángulos y la simetría presente.

En el espacio 3D, encontramos figuras como la esfera, el cilindro, el cono, el cubo, la pirámide, el prisma y el toro, entre otras. Estas figuras tienen una mayor variedad de características, como la presencia de caras, aristas y vértices, así como la posibilidad de calcular su volumen y área superficial.

El estudio de las figuras geométricas implica el análisis de sus propiedades geométricas, como el perímetro, el área, el volumen, la simetría y la congruencia. Además, se exploran conceptos como los ángulos, las rectas y los planos que intervienen en la definición y descripción de estas figuras.

La geometría es una herramienta poderosa que nos permite comprender y representar el mundo que nos rodea. Desde la construcción de edificios hasta la creación de modelos virtuales en la informática, el conocimiento de las figuras geométricas y sus propiedades es esencial para resolver problemas prácticos y desarrollar soluciones innovadoras.

En resumen, las figuras geométricas son elementos fundamentales en el estudio de la geometría y desempeñan un papel crucial en numerosos campos de aplicación. Su comprensión nos permite analizar y describir las propiedades espaciales de los objetos, así como desarrollar algoritmos y soluciones eficientes en el ámbito de la informática y otras disciplinas relacionadas.

## **Implementación de la estructura de control lógica secuencial**

## 13. Ortoedro

**Analizar, diseñar, evaluar e implementar un algoritmo para calcular el área, perímetro y diagonal de un Ortoedro.**

Imagen que contiene Diagrama

Descripción generada automáticamente

X = 2(ab + ac + bc)

V = a\*b\*c

D = √(a\*a + b\*b + c\*c)

**Código en C**

#include <stdio.h>

#include <math.h>

int area (int lado\_a, int lado\_b, int lado\_c);

int volumen (int lado\_a, int lado\_b, int lado\_c);

float diagonal(int lado\_a, int lado\_b, int lado\_c);

void main ()

{

int a, b, c;

float D, X, V;

a=b=c=0;

D=X=V=0.0;

printf ("Calcula el área, diagonal y volumen de un ortoedro \n");

printf ("Ingresa el lado a \n");

scanf ("%i",&a);

printf ("Ingresa el lado b \n");

scanf ("%i",&b);

printf ("Ingresa el lado c \n");

scanf ("%i",&c);

X = area(a,b,c);

V = volumen(a,b,c);

D = diagonal(a,b,c);

printf("El area del Ortoedro es %f \n", X);

printf("La diagonal del Ortoedro es %f \n", D);

printf("El volumen del Ortoedro es %f \n", V);

getch();

}

int area (int lado\_a, int lado\_b, int lado\_c)

{

int calcula;

calcula = 2\*((lado\_a\*lado\_b)+(lado\_a\*lado\_c)+(lado\_b\*lado\_c));

return  calcula;

}

int volumen (int lado\_a, int lado\_b, int lado\_c)

{

int calcula;

calcula = lado\_a \* lado\_b \* lado\_c;

return  calcula;

}

float diagonal (int lado\_a, int lado\_b, int lado\_c)

{

float calcula;

calcula = sqrt(pow(lado\_a,2) + pow(lado\_b,2) + pow(lado\_c,2));

return calcula;

}

## 14. Cono

**Analizar, diseñar, evaluar e implementar un algoritmo para calcular el área y volumen de un Cono.**

r = radio del cono

Diagrama

Descripción generada automáticamente con confianza bajaX = área del cono

V = volumen del cono

PI = 3.14

h = altura del cono

g = generatriz del cono

**Código en C**

#include <stdio.h>

#include <math.h>

float radio (int generatriz\_g, int altura\_h);

float area (float radio\_r, int generatriz\_g, float PI);

float volumen (float radio\_r, int altura\_h, float PI);

void main ()

{

int h, g;

float r, X, V, PI;

PI = 3.14;

h=g=0;

r=X=V=0.0;

printf ("Calcula el área y volumen de un cono \n");

printf ("Ingresa el valor de altura \n");

scanf ("%i",&h);

printf ("Ingresa el valor de generatriz \n");

scanf ("%i",&g);

r = radio(g,h);

X = area(r,g,PI);

V = volumen(r,h,PI);

printf("El radio del cono es %f \n", r);

printf("El area del cono es %f \n", X);

printf("El volumen del cono es %f \n", V);

getch();

}

float radio (int generatriz\_g, int altura\_h)

{

float calcula;

calcula = sqrt(pow(generatriz\_g,2)-(pow(altura\_h,2)));

return  calcula;

}

float area (float radio\_r, int generatriz\_g, float PI)

{

float calcula;

calcula = PI\*radio\_r\*(generatriz\_g+radio\_r);

return  calcula;

}

float volumen (float radio\_r, int altura\_h, float PI)

{

float calcula;

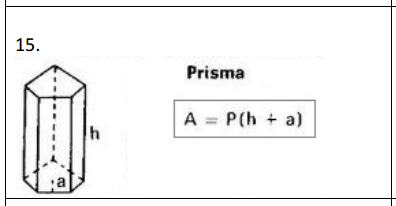
calcula = (PI\*(pow(radio\_r,2)\*altura\_h))/3;

return calcula;

}

## 15. Prisma

**Analizar, diseñar, evaluar e implementar un algoritmo para calcular el área y volumen de un Prisma pentagonal.**

P = perímetro del prisma

X = área del prisma

V = volumen del prisma

Ab = área de la base

l = lado del prisma

h = altura del prisma

a = apotema del prisma

**Código en C**

#include <stdio.h>

#include <math.h>

int perimetro (int lado\_l);

float areabase (int perimetro\_p, int apotema\_a);

int area (int perimetro\_p, int altura\_h, int apotema\_a);

float volumen(float areabase\_ab, int altura\_h);

void main ()

{

int l, h, a, P;

float Ab, X, V;

l=h=a=P= 0;

Ab=X=V=0.0;

printf ("Calcula el área y volumen de un prisma pentagonal \n");

printf ("Ingresa valor del lado \n");

scanf ("%i",&l);

printf ("Ingresa valor de la altura \n");

scanf ("%i",&h);

printf ("Ingresa valor de la apotema \n");

scanf ("%i",&a);

P = perimetro(l);

Ab = areabase(P,a);

X = area(P,h,a);

V = volumen(Ab,h);

printf("El perimetro del Prisma es %i \n", P);

printf("El area del Prisma es %f \n", X);

printf("El volumen del Prisma es %f \n", V);

getch();

}

int perimetro (int lado\_l)

{

int calcula;

calcula = lado\_l\*5;

return calcula;

}

float areabase (int perimetro\_p, int apotema\_a)

{

float calcula;

calcula = (perimetro\_p\*apotema\_a)/2;

return calcula;

}

int area (int perimetro\_p, int altura\_h, int apotema\_a)

{

int calcula;

calcula = perimetro\_p\*(altura\_h+apotema\_a);

return calcula;

}

float volumen (float areabase\_ab, int altura\_h)

{

float calcula;

calcula = areabase\_ab\*altura\_h;

return calcula;

}

## **Conclusión**

La implementación de la estructura de control lógica selectiva con funciones en lenguaje C es de gran importancia debido a la flexibilidad y modularidad que proporciona en el diseño y desarrollo de programas. Al utilizar funciones junto con estructuras de control selectivas, podemos escribir código más legible, mantenible y reutilizable.

Algunos puntos importantes:

1. Modularidad y reutilización: La combinación de estructuras de control selectivas con funciones nos permite dividir el código en bloques lógicos y reutilizables. Podemos encapsular la lógica de decisión en funciones específicas y llamarlas en diferentes partes del programa cuando sea necesario. Esto promueve el modularidad del código y facilita la reutilización de funciones en múltiples contextos.
2. Legibilidad y mantenibilidad: Al utilizar funciones con estructuras de control selectivas, el código se vuelve más legible y comprensible. Cada función puede tener un propósito claro y específico, lo que facilita la comprensión del flujo de ejecución. Además, si se requieren cambios o actualizaciones en la lógica de decisión, solo es necesario realizar modificaciones en una función específica, lo que mejora la mantenibilidad del código.
3. Separación de responsabilidades: El uso de funciones con estructuras de control selectivas permite separar diferentes responsabilidades en el programa. Podemos tener funciones especializadas para manejar diferentes casos o condiciones, lo que facilita la gestión y organización del código. Además, esta separación de responsabilidades favorece la identificación y solución de errores, ya que el código se divide en unidades más pequeñas y manejables.
4. Pruebas y depuración más sencillas: Al utilizar funciones con estructuras de control selectivas, las pruebas y depuración del código se vuelven más sencillas. Podemos realizar pruebas unitarias en cada función individualmente, verificando que la lógica de decisión funcione correctamente. Además, si se encuentra un error, la estructura modular nos permite aislar y solucionar el problema en una función específica sin afectar al resto del programa.

En resumen, la implementación de la estructura de control lógica selectiva con funciones en lenguaje C ofrece beneficios significativos en términos de modularidad, reutilización, legibilidad, mantenibilidad y separación de responsabilidades. Al utilizar esta combinación de conceptos, podemos escribir programas más estructurados, flexibles y fáciles de mantener. Además, facilita las pruebas y depuración del código, mejorando la calidad y confiabilidad del software desarrollado.