



Universidad Nacional Autónoma de México

Colegio de Ciencias y Humanidades Plantel Naucalpan

Sistemas Computacionales y Desarrollo de Software

Estructuras básicas del lenguaje C

Programa que calcula el Perímetro y Área de figuras geométricas

Hernández Espinoza Gabriel Emanuel

01/06/2023

Índice

Introducción.....	3
Justificación.....	4
Fundamentos de Programación en C: Enfoque Estructurado....	5
Estructuras básicas del Lenguaje C.....	6
Programa que Calcula el Area y Perimetro de diferentes figuras Geométricas.....	10
Conclusiones.....	20
Referencias	21

Introducción

Hoy en día, la programación es una parte fundamental de nuestra vida, cuando vas al banco, cuando compras algo en línea, el algoritmo de recomendación o sugerencia de amistades en tus redes sociales, los videos que te salen en youtube, crear un documento en word, las recomendaciones musicales que te hace spotify, etc. Todo es programación y es gracias a esta ciencia que muchísimas actividades que requieren mano de obra humana como la fabricación de autos se llegan a automatizar, reduciendo costos, riesgos así como facilitando y optimizando la elaboración de actividades y tareas complejas como repetitivas.

¿Pero qué es la programación?, ¿Son solamente instrucciones?, como su nombre lo indica, es el proceso de dar órdenes o pasos a seguir a una computadora para que esta realice tareas por nosotros, cosas complejas como hacer miles de operaciones y cálculos matemáticos que a un grupo de seres humanos les tomaría horas o incluso días realizar, a la computadora le tomaría solo segundos. Es gracias a la programación que podemos hacer predicciones a gran escala en cuestión de segundos, modelos matemáticos, gráficas, análisis de datos, etc. Pero la programación es más que solo instrucciones, ya que detrás de ella, sin una lógica, sin un sentido, sin un patrón u orden de pasos o lo que llamamos algoritmos, la programación no serviría de nada, por ejemplo, en algo tan sencillo como calcular la suma de dos números en una calculadora común y corriente se ocupa programación, ¿Pero como sabe calculadora que hacer exactamente?, aquí es cuando entra la lógica y el raciocinio humano que es el que le indica a la “computadora” o en este caso a la calculadora como hacer dichas operaciones.

Dicho todo esto, llegamos a la conclusión de que programar es un medio de comunicación entre un ser humano y una computadora, o incluso de una computadora a otra computadora llamándose también como lenguaje de programación, porque al igual que el Español, El inglés o incluso las Matemáticas que son consideradas un lenguaje, la programación también tiene una estructura, una sintaxis y un orden y es gracias a esto que podemos crear y hacer cosas tan complejas con una computadora, como el presente trabajo, el cual consiste en un programa que calcula el área y perímetro de múltiples figuras geométricas abarcando el modulo “estructuras básicas del lenguaje c” de la opción técnica de sistemas computacionales y desarrollo de software donde se hace el uso de ciclos if-else una condición o estructura sintáctica que le permite a la computadora tomar decisiones, o la instrucción switch, que nos permite hacer un menu en un programa permitiéndonos escoger qué tarea queremos realizar primero entre otras muchas.

Justificación

El presente trabajo busca demostrar todo lo que hay detrás de un software o como la mayoría de la gente conoce, aplicaciones, que en este caso, estudiantes de todo el mundo ocupan para realizar sus tareas de matemáticas, un ejemplo de dichas aplicaciones pueden ser Photomath, Mathway, WolframAlpha, Mathway etc, las cuales solo con ingresar el problema que deseamos resolver o los datos que se desean operar, en una fracción de segundos la computadora nos proporciona el resultado con dicho procedimiento, a diferencia de un humano que le demorara más tiempo realizar dichas actividades.

En Sistemas Computacionales y Desarrollo de Software en el módulo de Estructuras Básicas del lenguaje C se abarcaba la estructuración y sintaxis en en lenguaje C para la creación de programas básico que nos hace posible el correcto funcionamiento de un programa donde se busca aprender, saber, entender y dominar los conceptos de dicho módulo para evitar errores de código así como la elaboración de programas grandes y complejos.

Dicho trabajo, conciste en un programa que calcula el área y perímetro de una figura geométrica, dicha figura geométrica será elegida según la necesidad del usuario, así mismo, los datos serán dados por el usuario. ¿Pero porque este programa tiene necesariamente dichas condiciones o realiza estas tareas en lugar de que solo se calcule el área o el perímetro de solo una figura?, ¿Porque los datos de dicha figura deben ser ingresados por un usuario en lugar de predisponer unos?. La respuesta a esto es analizar, entender y explicar el funcionamiento de dichos programas que el celular de un alumno cualquiera tendría descargado para realizar sus tareas de matemáticas. podría decirse que esta actividad incluso se ha convertido una parte esencial o fundamental de la vida de un estudiante, el problema es que muchas personas no saben o hacen el intento de entender cómo funcionan dichos software que les facilitan la vida. Tomando en cuenta el crecimiento y avance exponencial de la tecnología y la increíble rapidez que esta es introducida en la vida cotidiana de una persona promedio, considero que al menos es fundamental que las personas que hacen uso de dichas herramientas tengan al menos una noción de cómo es que dichas aplicaciones funcionan.

Fundamentos de Programación en C: Enfoque Estructurado

Como se mencionó en la Introducción de este trabajo, programar consiste en dar una serie estructurada de órdenes o instrucciones a una computadora, a dicho conjunto de instrucciones se le conoce como código, donde un código es un método de representación de la información, en el caso de un programa, dicha información son las instrucciones que conforman dicho código, dicha información o datos, representan algo y ese algo la computadora los interpreta con 0 y 1 o conocido como el Código Binario. Debido a que al aprender a programar C, se debe aprender a usar el código binario, C es considerado un lenguaje de bajo nivel o dicho en otras palabras es un lenguaje que está “más próximo al software”.

Existen tres estilos de programación, la programación estructurada y desestructurada, en este caso solo nos enfocaremos en la programación estructurada, la cual es un enfoque de programación que se basa en el principio de dividir un programa en estructuras de control más pequeñas y manejables. Se centra en la organización lógica y clara del código fuente, utilizando principalmente tres estructuras de control: secuencia, selección y repetición.

1. Secuencia: Las instrucciones se ejecutan en orden secuencial, una tras otra. Es la forma más básica de control de flujo y permite que las acciones se realicen en un orden predeterminado.

2. Selección: Se utilizan estructuras de control condicionales, como las declaraciones "if" (si) y "switch" (selección múltiple), para tomar decisiones basadas en condiciones. Estas estructuras permiten ejecutar diferentes bloques de código dependiendo de si se cumple o no una condición específica.

3. Repetición: Se utilizan estructuras de control iterativas, como los bucles "for" (para) y "while" (mientras), para repetir un bloque de código varias veces hasta que se cumpla una condición de terminación.

La programación estructurada busca evitar el uso de estructuras de control complejas y saltos incondicionales, como los "goto", que pueden dificultar la comprensión y el mantenimiento del código. En cambio, fomenta la modularidad y la claridad, dividiendo el programa en pequeñas unidades funcionales llamadas subrutinas o funciones.

La programación estructurada fue popularizada en la década de 1960 por Edsger Dijkstra y otros pioneros de la informática. Se considera una forma más legible y fácil de mantener el código en comparación con enfoques más antiguos, como la programación no estructurada.

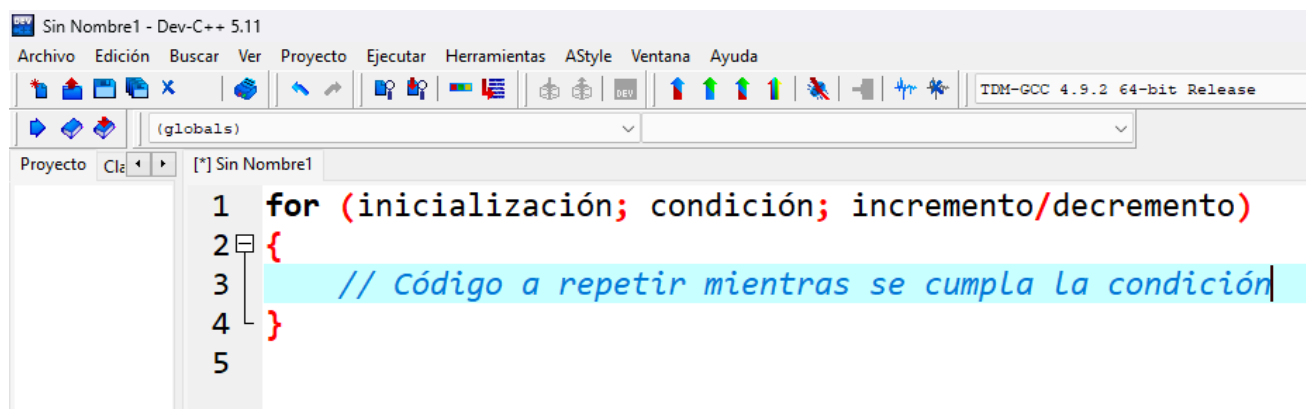
Estructuras básicas del Lenguaje C

En C, existen varias estructuras de control de bucles para implementar ciclos en un programa. Estas estructuras permiten repetir un bloque de código varias veces hasta que se cumpla una condición de terminación. Las tres estructuras de ciclos más comunes en C son el ciclo for, if, if-else, while, do while, switch, entre otros más.

Para dicho proyecto, se utilizaron los ciclos if-else, do-while y switch, aunque el resto de los ciclos no se hayan usado, considero importante hablar de ellas y explicarlas porque son las primeras que se enseña al aprender a programar así como de lo más fundamental.

1. Bucle "for":

El bucle "for" se utiliza cuando se conoce el número exacto de iteraciones que se deben realizar, sirve para realizar una cierta cantidad de veces una tarea, su sintaxis es la siguiente:

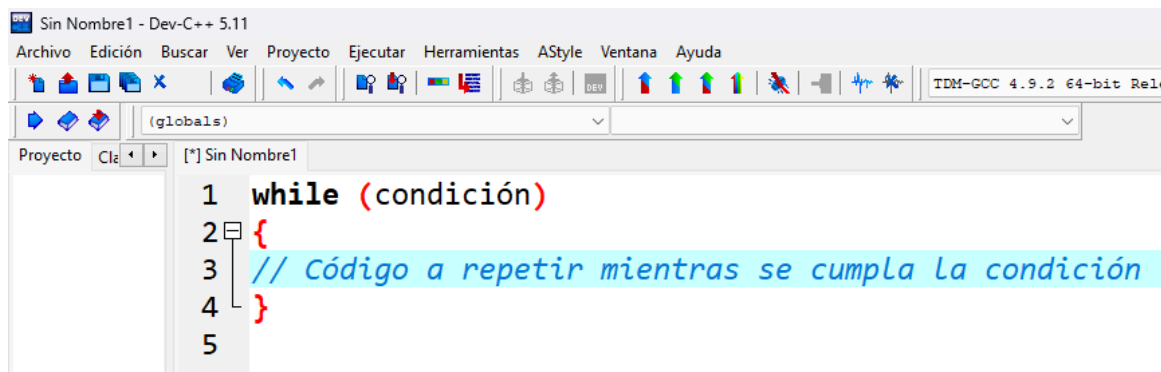


```
1 for (inicialización; condición; incremento/decremento)
2 {
3     // Código a repetir mientras se cumpla la condición
4 }
5
```

La inicialización se realiza antes de que comience el bucle y generalmente se utiliza para declarar e inicializar variables de control. La condición se evalúa antes de cada iteración y, si es verdadera, el bloque de código se ejecuta. Después de cada iteración, se actualizan las variables de control. Si la condición es falsa, el bucle se termina.

2. Bucle "while":

El bucle "while" se utiliza cuando la condición de terminación no se conoce de antemano y se evalúa al comienzo de cada iteración. Tiene la siguiente sintaxis:

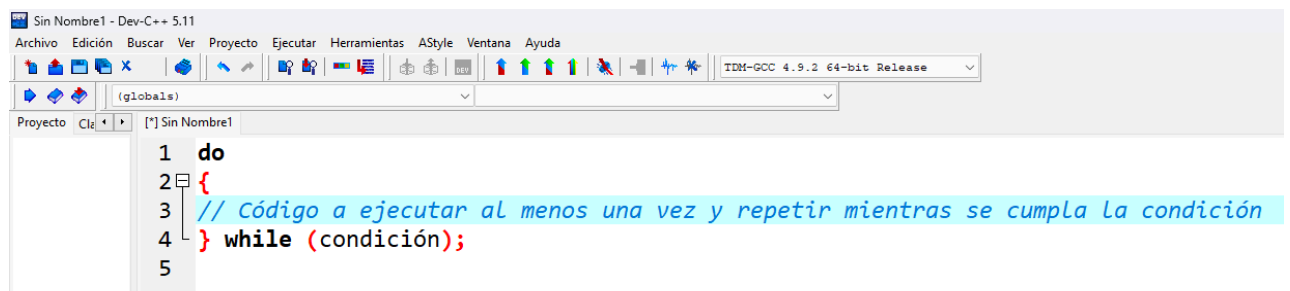


```
1 while (condición)
2 {
3 // Código a repetir mientras se cumpla la condición
4 }
5
```

El bloque de código se ejecuta siempre que la condición sea verdadera. Si la condición es falsa, el bucle se detiene y el programa continúa con la siguiente instrucción después del bucle.

3. Bucle "do-while":

El bucle "do-while" también se utiliza cuando la condición de terminación no se conoce de antemano, pero a diferencia del bucle "while", la condición se evalúa al final de cada iteración. Esto garantiza que el bloque de código se ejecute al menos una vez. Tiene la siguiente sintaxis:

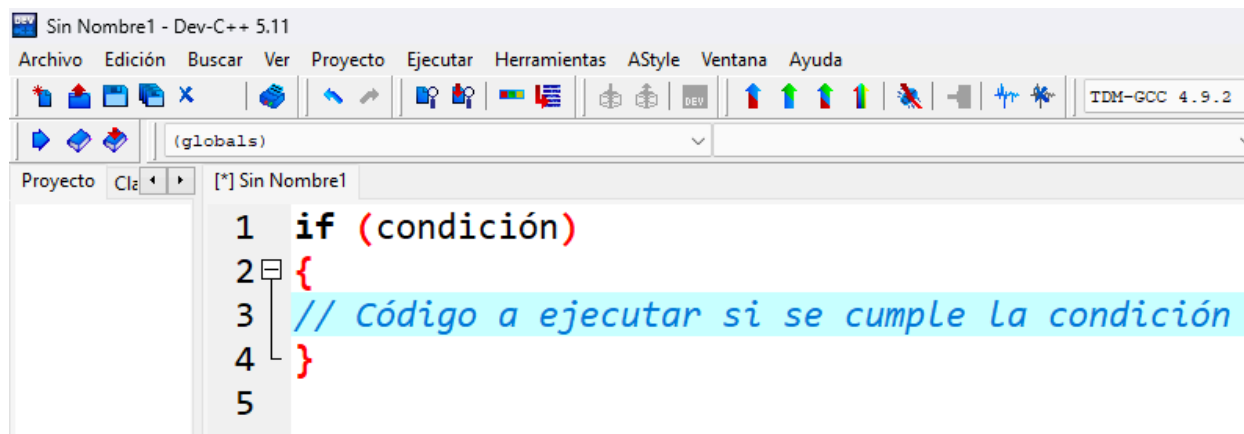


```
1 do
2 {
3 // Código a ejecutar al menos una vez y repetir mientras se cumpla la condición
4 } while (condición);
5
```

El bloque de código se ejecuta y luego la condición se evalúa. Si la condición es verdadera, el bucle se repite. Si es falsa, el bucle se termina y el programa continúa con la siguiente instrucción después del bucle.

Estas estructuras de control de bucles proporcionan diferentes formas de implementar ciclos en C, y la elección de la estructura adecuada depende de los requisitos y lógica específicos del programa.

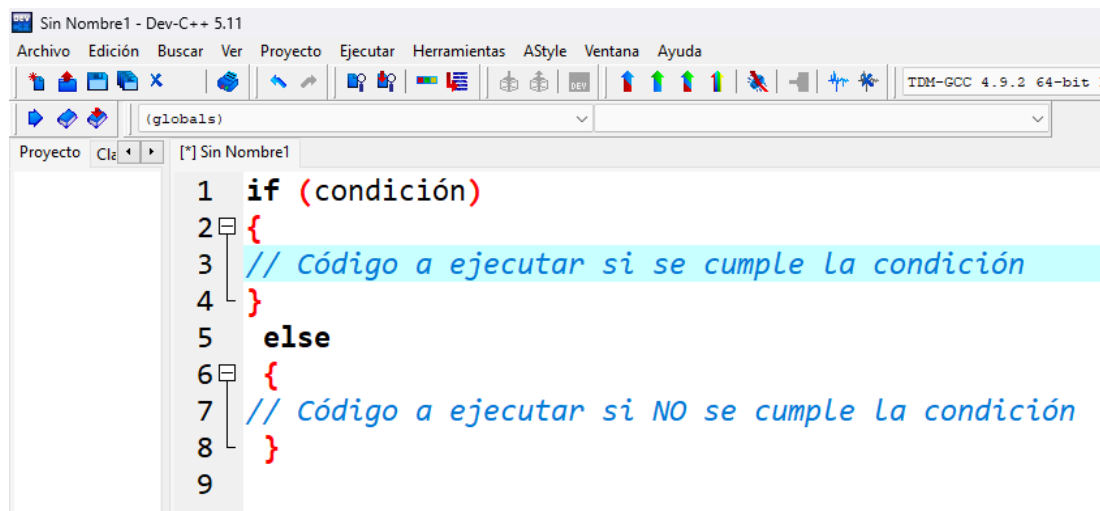
4. If. El "if" en C se utiliza para ejecutar un bloque de código si se cumple una condición específica. La sintaxis es la siguiente:

A screenshot of the Dev-C++ 5.11 IDE. The title bar reads "Sin Nombre1 - Dev-C++ 5.11". The menu bar includes "Archivo", "Edición", "Buscar", "Ver", "Proyecto", "Ejecutar", "Herramientas", "AStyle", "Ventana", and "Ayuda". The toolbar contains various icons for file operations, editing, and execution. The status bar at the bottom indicates "TDM-GCC 4.9.2". The main editor window shows a C++ code snippet:

```
1  if (condición)
2  {
3      // Código a ejecutar si se cumple la condición
4  }
5
```

Si la condición es verdadera, el bloque de código dentro del "if" se ejecuta. Si la condición es falsa, el bloque se omite y el programa continúa con la siguiente instrucción después del "if".

2. **if-else:** El "if-else" se utiliza cuando se desea ejecutar un bloque de código si se cumple una condición y otro bloque de código si no se cumple. La sintaxis es la siguiente:

A screenshot of the Dev-C++ 5.11 IDE, similar to the first one. The main editor window shows a C++ code snippet for an if-else statement:

```
1  if (condición)
2  {
3      // Código a ejecutar si se cumple la condición
4  }
5  else
6  {
7      // Código a ejecutar si NO se cumple la condición
8  }
9
```

Si la condición es verdadera, se ejecuta el bloque de código dentro del "if". Si la condición es falsa, se ejecuta el bloque de código dentro del "else".

3. Switch. El "switch" se utiliza cuando se desea realizar una selección entre varias opciones diferentes. Se evalúa una expresión y se compara con cada caso para determinar cuál de ellos coincide. La sintaxis es la siguiente:


```
1 switch (expresión)
2 {
3     case valor1:
4         // Código a ejecutar si la expresión coincide con valor1
5         break;
6     case valor2:
7         // Código a ejecutar si la expresión coincide con valor2
8         break;
9     // ...
10    default:
11        // Código a ejecutar si la expresión no coincide con ningún caso anterior
12    }
13
```

El bloque de código asociado a cada caso se ejecutará si la expresión coincide con el valor correspondiente. Si no se encuentra ninguna coincidencia, se ejecuta el bloque de código dentro de "default" (opcional).

Programa que Calcula el Area y Perimetro de diferentes figuras Geometricas

Una vez explicados y aclarados todas los conceptos, definiciones y bases necesarias para poder entender por completo este proyecto, procederemos a ver, analizar y explicar dicho trabajo, el código de dicho trabajo es el siguiente, en donde cada fragmento del código será explicado con sumo detalle

```
1  #include<stdio.h>
2  int main()
3  {
4      int opcion;
5      int perimetro, area, lado, base, altura, Dmayor, dmenor, diametro, radio;
6      double pi = 3.141516;
7      printf("\t\tPrograma que calcula el area y perimetro de figuras geometricas\n\n");
8      printf("1. Area y Perimetro de un Cuadrado");
9      printf("\n2. Area y Perimetro de un Rectangulo");
10     printf("\n3. Area y Perimetro de un paralelogramo");
11     printf("\n4. Area y Perimetro de un Rombo");
12     printf("\n5. Area y Perimetro de un Triangulo");
13     printf("\n6. Area y Perimetro de un Circulo\n\n");
14     printf("Opcion:");
15     scanf("%d",&opcion);
16
17     switch(opcion)
18     {
19         case 1:
20             printf("\t\t\tCalcular el Area y Perimetro de un cuadrado\n\n");
21             printf("Ingresa uno de los lados del cuadrado: ");
22             do
23             {
24                 scanf("%d",&lado);
25                 if(lado <= 0)
26                 {
27                     printf("Error!, vuelve a ingresar los datos:");
28                 }
29             } while(lado <= 0);
```

Como librería sólo se ocupará la librería stdio.h (standar input output), que es la que se encarga de proporcionarnos lo que son las funciones printf, y scanf, fundamentales para poder mostrar mensajes en pantalla así como recibir datos proporcionados por el usuario.

```
1  #include<stdio.h>
2  int main()
3  {
```

Posteriormente se produce a declarar variables de tipo entero o flotantes, por casos prácticos, en dicho trabajo las variables se declararon como enteras. La variable “int opción” nos permitirá el uso y manipulación de la estructura “switch”, mientras que el resto de variables nos permitirán el uso y almacenamiento de datos que serán solicitados para el cálculo del área y del perímetro de la figura geométrica que se

escoja. Todo esto dentro de la función `int main`, que es la función principal en la que se realizará todo el procedimiento de ejecución del código .

```
1  #include<stdio.h>
2  int main()
3  {
4      int opcion;
5      int perimetro, area, lado, base, altura, Dmayor,dmenor, diametro, radio;
6      double pi = 3.141516;
```

A excepción de las variables enteras, la variable “pi” será la única que se dejará como un dato de tipo doble debido a que pi es una cantidad infinita de números, recordemos que para el almacenamiento de números grandes nos conviene el uso del tipo de dato `double`.

Posteriormente se procede a poner las los mensajes que se mostraran en pantalla a la hora de ejecutar dicho programa, donde con el uso de una variable “scanf” que se muestra en la siguiente imagen, nos permitirá introducir qué opción queremos realizar.

```
1  #include<stdio.h>
2  int main()
3  {
4      int opcion;
5      int perimetro, area, lado, base, altura, Dmayor,dmenor, diametro, radio;
6      double pi = 3.141516;
7      printf("\t\tPrograma que calcula el area y perimetro de figuras geometricas\n\n");
8      printf("1. Area y Perimetro de un Cuadrado");
9      printf("\n2. Area y Perimetro de un Rectangulo");
10     printf("\n3. Area y Perimetro de un paralelogramo");
11     printf("\n4. Area y Perimetro de un Rombo");
12     printf("\n5. Area y Perimetro de un Triangulo");
13     printf("\n6. Area y Perimetro de un Circulo\n\n");
14     printf("Opcion:");
15     scanf("%d",&opcion);
16 }
```

Con el uso de la estructura “Switch”, según la opción que se haya escogido, se procederán hacer los cálculos correspondientes, para cada opción, la explicación de dicha parte es la siguiente :

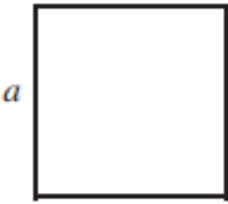
Opción 1: Area y Perimetro de un Cuadrado

Para el caso 1, donde se calcula el área y perímetro de un cuadrado se mostraran en pantalla los mensajes que se almacenan en cada “printf” que se encargaran de resaltar la opción que se eligió.

El uso de la estructura “if-else” nos sirve para que en caso de que se introduzca un número negativo nos muestre un error en pantalla, y junto con la instrucción “do-while” dicho mensaje de error se seguirá mostrando hasta que el usuario introduzca un número positivo, ya que no existen distancia negativas, por ende cualquier figura geométrica no puede tener valores negativos.

Pero en caso de que se introduzca un número positivo se procederá a realizar los cálculos correspondientes para el área y perímetro de un cuadrado, recordemos que para el área y perímetro de un cuadrado es:

Cuadrado



Perímetro: $P = 4a$
Área: $A = a^2$

Márquez, A. A. (2015). *Matemáticas simplificadas*

Finalmente, la instrucción “break” detendrá la ejecución del código finalizando el programa, evitando que se ejecute el resto de instrucciones u opciones que no se llevaron a cabo.

```

16
17
18 switch(opcion)
19 {
20     case 1:
21         printf("\t\t\tCalcular el Area y Perimetro de un cuadrado\n\n");
22         printf("Ingresa uno de los lados del cuadrado: ");
23         do
24         {
25             scanf("%d",&lado);
26             if(lado <= 0)
27             {
28                 printf("Error!, vuelve a ingresar los datos:");
29             }
30         } while(lado <= 0);
31         perimetro = 4*lado;
32         area = lado*lado;
33         printf("El perimetro del cuadrado es: %d\nEl area del cuadrado es: %d",perimetro,area);
34         break;

```

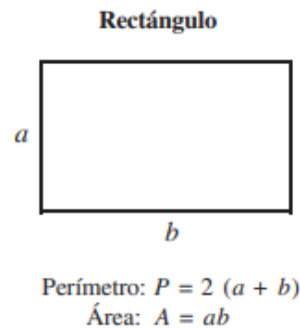
Opción 2: Área y Perímetro de un Rectángulo

Para el caso 2, donde se calcula el área y perímetro de un rectángulo se mostraran en pantalla los mensajes que se almacenan en cada “printf” que se encargan de resaltar la opción que se eligió.

El uso de la estructura “if-else” nos sirve para que en caso de que se introduzca un número negativo nos muestre un error en pantalla, y junto con la instrucción “do-while” dicho mensaje de error se seguirá mostrando hasta que el usuario

introduzca un número positivo, ya que no existen distancia negativas, por ende cualquier figura geométrica no puede tener valores negativos.

Pero en caso de que se introduzca un número positivo se procederá a realizar los cálculos correspondientes para el área y perímetro de un cuadrado, recordemos que para el area y perímetro de un rectángulo es:



Márquez, A. A. (2015). *Matemáticas simplificadas*

Finalmente, la instrucción “break” detendrá la ejecución del código finalizando el programa, evitando que se ejecute el resto de instrucciones u opciones que no se llevaron a cabo.

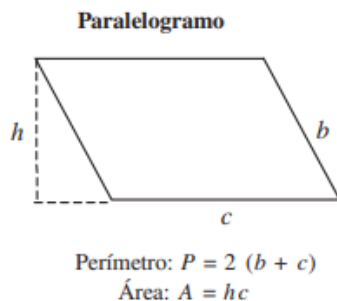
```
35 | case 2:
36 |     printf("\t\t\tCalcular el Area y Perimetro de un rectangulo\n\n");
37 |     printf("Ingresa un de los lado del rectangulo:");
38 |     do
39 |     {
40 |         scanf("%d",&lado);
41 |         if(lado <= 0)
42 |         {
43 |             printf("Error!, vuelve a ingresar los datos:");
44 |         }
45 |     }
46 |     while(lado <= 0);
47 |     printf("Ingresa la base del rectangulo:");
48 |     do
49 |     {
50 |         scanf("%d",&base);
51 |         if(base <= 0 )
52 |         {
53 |             printf("Error!, vuelve a ingresar los datos:");
54 |         }
55 |     }
56 |     while(base <= 0);
58 |     perimetro = 2*(lado + base);
59 |     area = lado*base;
60 |     printf("El perimetro del rectangulo es: %d\nEl area del rectangulo es: %d",perimetro,area);
61 |     break;
```

Opción 3: Area y Perimetro de un Paralelogramo

Para el caso 3, donde se calcula el área y perímetro de un paralelogramo se mostraran en pantalla los mensajes que se almacenan en cada “printf” que se encargan de resaltar la opción que se eligió.

El uso de la estructura “if-else” nos sirve para que en caso de que se introduzca un número negativo nos muestre un error en pantalla, y junto con la instrucción “do-while” dicho mensaje de error se seguirá mostrando hasta que el usuario introduzca un número positivo, ya que no existen distancia negativas, por ende cualquier figura geométrica no puede tener valores negativos.

Pero en caso de que se introduzca un número positivo se procederá a realizar los cálculos correspondientes para el área y perímetro de un cuadrado, recordemos que para el area y perimetro de un paralelogramo es:



Márquez, A. A. (2015). *Matemáticas simplificadas*

Finalmente, la instrucción “break” detendrá la ejecución del código finalizando el programa, evitando que se ejecute el resto de instrucciones u opciones que no se llevaron a cabo.

```
62 case 3:
63     printf("\t\t\t\t\tCalcular el Area y Perimetro de un paralelogramo\n\n");
64     printf("Ingresa uno de los lados del paralelogramo: ");
65     do
66     {
67         scanf("%d",&lado);
68         if(lado <= 0)
69         {
70             printf("Error!, vuelve a ingresar los datos:");
71         }
72     }
73     while(lado <= 0);
74     printf("Ingresa la base del paralelogramo: ");
75     do
76     {
77         scanf("%d",&base);
78         if(base <= 0)
79         {
80             printf("Error!, vuelve a ingresar los datos:");
81         }
82     }
83     while(base <= 0);
84     printf("Ingresa la altura del paralelogramo: ");
85     do
```

```

86 | {
87 |     scanf("%d",&altura);
88 |     if(altura <= 0)
89 |     {
90 |         printf("Error!, vuelve a ingresar los datos:");
91 |     }
92 | }
93 | while(altura <= 0);
94 |     perimetro = 2*(base+lado);
95 |     area = altura*base;
96 |     printf("El perimetro del paralelogramo es: %d\nEl area del paralelogramo es: %d",perimetro,area);
97 |     break;

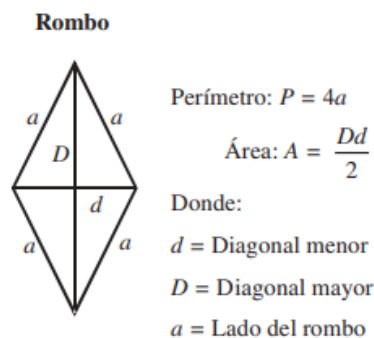
```

Opción 4: Area y Perimetro de un Rombo

Para el caso 4, donde se calcula el área y perímetro de un rombo se mostraran en pantalla los mensajes que se almacenan en cada “printf” que se encargan de resaltar la opción que se eligió.

El uso de la estructura “if-else” nos sirve para que en caso de que se introduzca un número negativo nos muestre un error en pantalla, y junto con la instrucción “do-while” dicho mensaje de error se seguirá mostrando hasta que el usuario introduzca un número positivo, ya que no existen distancia negativas, por ende cualquier figura geométrica no puede tener valores negativos.

Pero en caso de que se introduzca un número positivo se procederá a realizar los cálculos correspondientes para el área y perímetro de un cuadrado, recordemos que para el area y perimetro de un rombo es:



Márquez, A. A. (2015). *Matemáticas simplificadas*

Finalmente, la instrucción “break” detendrá la ejecución del código finalizando el programa, evitando que se ejecute el resto de instrucciones u opciones que no se llevaron a cabo.

```

98      case 4:
99          printf("\t\t\tCalcular el Area y Perimetro de un Rombo\n\n");
100         printf("Ingresa uno de los lados del rombo: ");
101         do
102         {
103             scanf("%d",&lado);
104             if(lado <= 0)
105             {
106                 printf("Error!, vuelve a ingresar los datos:");
107             }
108         }
109         while(lado <= 0);
110         printf("Ingresa la diagonal mayor del rombo: ");
111         do
112         {
113             scanf("%d",&Dmayor);
114
115             if(Dmayor <= 0)
116             {
117                 printf("Error!, vuelve a ingresar los datos:");
118             }
119         }
120         while(Dmayor <= 0);
121         printf("Ingresa la diagonal menor del rombo: ");
122         do
123         {
124             scanf("%d",&dmenor);
125             if(dmenor <= 0)
126             {
127                 printf("Error!, vuelve a ingresar los datos:");
128             }
129         }
130         while(dmenor <= 0);
131         perimetro = 4*lado;
132         area = (Dmayor*dmenor)/2;
133         printf("El perimetro del rombo es: %d\nEl area del rombo es: %d",perimetro,area);
134         break;

```

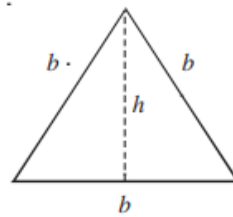
Opción 5: Area y Perimetro de un Triángulo

Para el caso 5, donde se calcula el área y perímetro de un triángulo se mostraran en pantalla los mensajes que se almacenan en cada "printf" que se encargan de resaltar la opción que se eligió.

El uso de la estructura "if-else" nos sirve para que en caso de que se introduzca un número negativo nos muestre un error en pantalla, y junto con la instrucción "do-while" dicho mensaje de error se seguirá mostrando hasta que el usuario introduzca un número positivo, ya que no existen distancia negativas, por ende cualquier figura geométrica no puede tener valores negativos.

Pero en caso de que se introduzca un número positivo se procederá a realizar los cálculos correspondientes para el área y perímetro de un cuadrado, recordemos que para el area y perimetro de un triángulo es:

Equilátero



$$\text{Perímetro: } P = 3b$$

$$\text{Área: } A = \frac{bh}{2}$$

Márquez, A. A. (2015). *Matemáticas simplificadas*

Finalmente, la instrucción “break” detendrá la ejecución del código finalizando el programa, evitando que se ejecute el resto de instrucciones u opciones que no se llevaron a cabo.

```
134 case 5:
135     printf("\t\t\tCalcular el Area y Perimetro de un triangulo\n\n");
136     printf("Ingresa un de los lado del triangulo:");
137     do
138     {
139         scanf("%d",&lado);
140         if(lado <= 0)
141         {
142             printf("Error!, vuelve a ingresar los datos:");
143         }
144     }
145     while(lado <= 0);
146     printf("Ingresa la altura del rectangulo:");
147     do
148     {
149         scanf("%d",&altura);
150         if(altura <= 0 )
151         {
152             printf("Error!, vuelve a ingresar los datos:");
153         }
154     }
155     while(altura <= 0);
156
157     perimetro = 3*lado;
158     area = (lado*altura)/2;
159     printf("El perimetro del triangulo es: %d\nEl area del triangulo es: %d",perimetro,area);
160
161     break;
```

Opción 6: Area y Perimetro de un Círculo

Para el caso 5, donde se calcula el área y perímetro de un triángulo se mostraran en pantalla los mensajes que se almacenan en cada “printf” que se encargan de resaltar la opción que se eligió.

El uso de la estructura “if-else” nos sirve para que en caso de que se introduzca un número negativo nos muestre un error en pantalla, y junto con la instrucción

“do-while” dicho mensaje de error se seguirá mostrando hasta que el usuario introduzca un número positivo, ya que no existen distancia negativas, por ende cualquier figura geométrica no puede tener valores negativos.

Pero en caso de que se introduzca un número positivo se procederá a realizar los cálculos correspondientes para el área y perímetro de un cuadrado, recordemos que para el area y perimetro de un triángulo es:

Perímetro

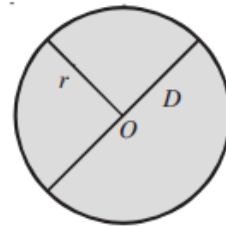
$$P = 2\pi r = D\pi$$

Donde:

r = Radio, D = Diámetro y $\pi = 3.14159...$

Área

$$A = \pi r^2 = \frac{1}{4} \pi D^2$$



Márquez, A. A. (2015). *Matemáticas simplificadas*

Finalmente, la instrucción “break” detendrá la ejecución del código finalizando el programa, evitando que se ejecute el resto de instrucciones u opciones que no se llevaron a cabo.

```

162 |
163 |         case 6:
164 |             printf("\t\t\tCalcular el Area y Perimetro de un circulo\n\n");
165 |             printf("Ingresa el diametro del circulo:");
166 |             do
167 |             {
168 |                 scanf("%d",&diametro);
169 |                 if(diametro <= 0)
170 |                 {
171 |                     printf("Error!, vuelve a ingresar los datos:");
172 |                 }
173 |             } while(diametro <= 0);
174 |             printf("Ingresa el radio del circulo:");
175 |             do
176 |             {
177 |                 scanf("%d",&radio);
178 |                 if(radio <= 0 )
179 |                 {
180 |                     printf("Error!, vuelve a ingresar los datos:");
181 |                 }
182 |             } while(radio <= 0);
183 |             perimetro = pi * diametro;
184 |             area = pi*(radio^2);
185 |             printf("El perimetro del circulo es: %d\nEl area del circulo es: %d",perimetro,area);
186 |             break;
187 |         default:
188 |             printf("Opcion no valida");
189 |     }
190 |     return 0;
191 | }

```

Finalmente, en caso de que se introduzca un número mayor al de las opciones disponibles, “default” mostrará un mensaje en pantalla indicándonos esto hasta que se introduzca una opción válida.

Conclusión

Dicho proyecto introdujo el tema de la programación y su importancia en la vida cotidiana, destacando cómo la automatización y la optimización de tareas complejas han sido posibles gracias a esta ciencia. Se explicó que la programación es un medio de comunicación entre humanos y computadoras, donde las instrucciones dadas a una computadora deben tener lógica y un patrón de pasos llamado **algoritmo** para que sean útiles.

Este trabajo también presentó la justificación de un proyecto que busca enseñar y comprender los conceptos básicos de programación a través de un programa que calcula el área y perímetro de diferentes figuras geométricas. Se resaltó la importancia de entender cómo funcionan las aplicaciones y herramientas tecnológicas que se utilizan en la vida diaria.

Además, se proporcionaron fundamentos de programación estructurada en C, explicando las estructuras de control como "for", "while", "do-while", "if", "if-else" y "switch", que son utilizadas en el programa mencionado para calcular las áreas y perímetros.

En resumen, se enfatiza la importancia de la programación en la vida moderna y la necesidad de comprender sus fundamentos para aprovechar y utilizar eficientemente la tecnología que nos rodea. El proyecto propuesto busca brindar una base sólida para aquellos interesados en aprender a programar y comprender cómo funcionan las aplicaciones que utilizan en su vida diaria.

Referencias

Vozmediano, A. M. (2017). *Aprender a programar en C: de 0 a 99 en un solo libro: Un Viaje Desde la Programación Estructurada en Pseudocódigo Hasta Las Estructuras de Datos Avanzadas en Lenguaje C.*

Márquez, A. A. (2015). *Matemáticas simplificadas.*