

Programacion estructurada.

Daniel Reyes Barrera

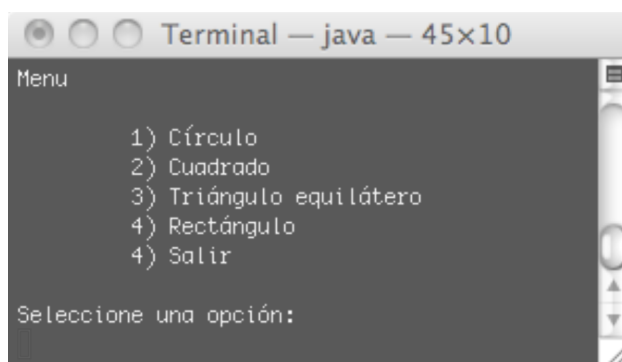
23 de noviembre de 2020

Resumen

En este documento se han resuelto algunos problemas computacionales utilizando las herramientas aprendidas en la clase 5 – Programación estructurada del curso de programación C++, como son Modularización, métodos y funciones las cuales nos proporcionan un mejor uso para un bloque recurrente de instrucciones.

1. Ejercicio 1.

Escriba un programa que imprima un menú para seleccionar un tipo de figura geométrica de la siguiente forma:



El usuario debe seleccionar una opción y el programa debe calcular el área y perímetro de la opción seleccionada. Programe cada opción en un método independiente. El programa debe regresar al menu principal hasta que el usuario seleccione la opción salir.

1.1. Problema computacional.

Objetivo: Calcular áreas de figuras deseadas de manera indefinida.

Entrada: Un número que represente la figura deseada y posteriormente sus dimensiones.

Salida: El area de la figura deseada.

1.2. Algoritmo.

Para solucionar el problema computacional, programamos el algoritmo dentro del ciclo de repetición **do-while** para que se siga ejecutando hasta que el usuario decida salir. Cada opción del calculo de áreas se programo métodos diferentes.

El código fuente está disponible en mi repositorio de git hub. [1]

1.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Se calcularon las siguientes áreas: Un circulo de radio 4 y el área de un rectangulo de base = 2 y altura = 5. . La salida del programa se observa en la Figura 1.

```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/calcular_area
Menu
1) Círculo
2) Cuadrado
3) Triángulo equilátero
4) Rectángulo
5) Salir
Seleccione una opción: 1
Radio: 4
Area = 50.2655
Menu
1) Círculo
2) Cuadrado
3) Triángulo equilátero
4) Rectángulo
5) Salir
Seleccione una opción: 4
Base: 2
Altura: 5
Area = 10
Menu
1) Círculo
2) Cuadrado
3) Triángulo equilátero
4) Rectángulo
5) Salir
Seleccione una opción: 5
Saliendo...
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 1: Ejecución de algunas instancias del problema.

2. Ejercicio 2.

El máximo común divisor de dos enteros es el entero más grande que puede dividir a cada uno de los dos números. Programe el algoritmo para calcular el máximo común divisor en un método independiente.

2.1. Problema computacional.

Objetivo: Dado dos número enteros calcular el máximo comun divisor.

Entrada: Dos números enteros.

Salida: El valor de MCD de ambos números introducidos.

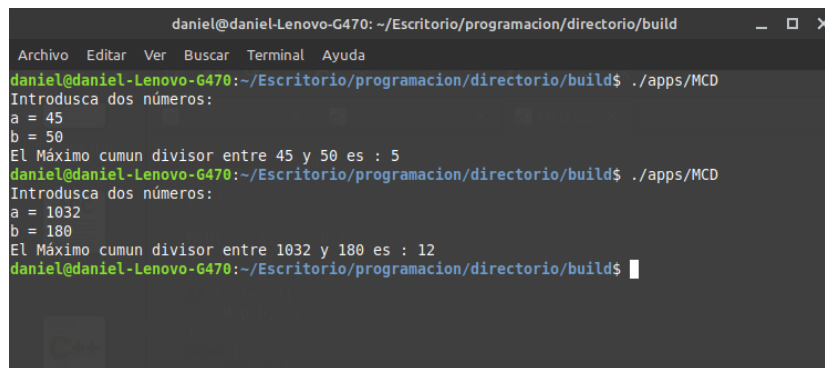
2.2. Algoritmo.

Para solucionar el problema computacional, utilizaremos el algoritmo de Euclides [6] que consiste en tomar en cuenta la parte entera de la división de los números (cociente) y su parte residual (resto), si el resto no es cero volvemos a dividir el divisor entre el cociente, aplicamos ese método hasta que el resto sea igual a 0 y entonces el ultimo divisor es el MCD.

El código fuente está disponible en mi repositorio de git hub. [2]

2.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: $a = 45$, $b = 50$ y $a = 1032$, $b = 180$. La salida del programa se observa en la Figura 2.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/MCD
Introduzca dos números:
a = 45
b = 50
El Máximo comun divisor entre 45 y 50 es : 5
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/MCD
Introduzca dos números:
a = 1032
b = 180
El Máximo comun divisor entre 1032 y 180 es : 12
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 2: Ejecución de algunas instancias del problema.

3. Ejercicio 3.

Programa el algoritmo para determinar si el número es palíndromo y el algoritmo para validar la entrada en métodos independientes. Sugerencia: Haga uso de los operadores módulo y división para separar el número tecleado en unidades, decenas, centenas, etc.

3.1. Problema computacional.

Objetivo: Dado un número entero determinar si es palíndromo.

Entrada: Un número entero mayor que 0.

Salida: La respuesta de si el número dado es o no palíndromo.

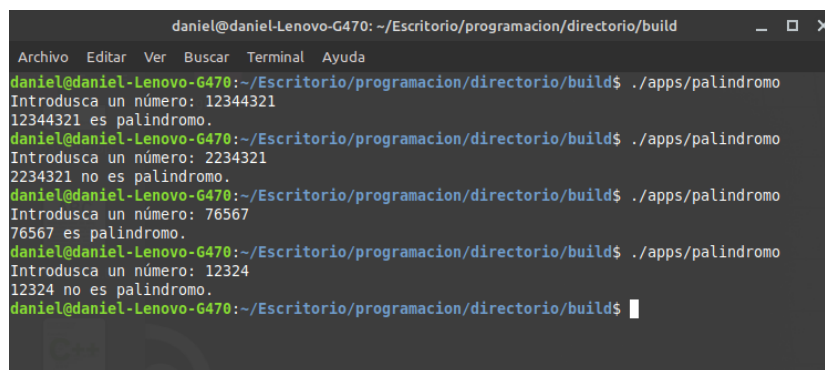
3.2. Algoritmo.

Para solucionar el problema computacional, utilizamos una variable de apoyo para comparar el número dado mediante la separación en unidades, decenas, centenas, etc. Utilizando los operadores módulo y división.

El código fuente está disponible en mi repositorio de git hub. [3]

3.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: 12344321, 2234321, 76567 y 12324. La salida del programa se observa en la Figura 3.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/palindromo
Introduzca un número: 12344321
12344321 es palindromo.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/palindromo
Introduzca un número: 2234321
2234321 no es palindromo.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/palindromo
Introduzca un número: 76567
76567 es palindromo.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/palindromo
Introduzca un número: 12324
12324 no es palindromo.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 3: Ejecución de algunas instancias del problema.

4. Ejercicio 4.

Programe el algoritmo de Schrage para generar números pseudo-aleatorios entre 0 y m . El j -ésimo elemento de la sucesión de números pseudo-aleatorios, denotado por I_j , se calcula con la siguiente ecuación:

$$I_j = a(I_{j-1} \bmod m)$$

$$I_j = \begin{cases} a(I_{j-1} \bmod q) - r[I_{j-1}/q] & \text{si } a(I_{j-1} \bmod q) - r[I_{j-1}/q] > 0 \\ a(I_{j-1} \bmod q) - r[I_{j-1}/q] + m & \text{en otro caso} \end{cases}$$

donde $a = 75$, $m = 2^{31} - 1$, $q = 127773$, $r = 2836$ y I_0 es la semilla del generador, cuyo valor se deja a criterio del programador. Los corchetes indican que el resultado de la división se trunca para obtener solamente la parte entera.

4.1. Problema computacional.

Objetivo: Calcular una cantidad n de números aleatorios dados por el usuario.

Entrada: Un número entero mayor que 0.

Salida: Una sucesión de números aleatorios.

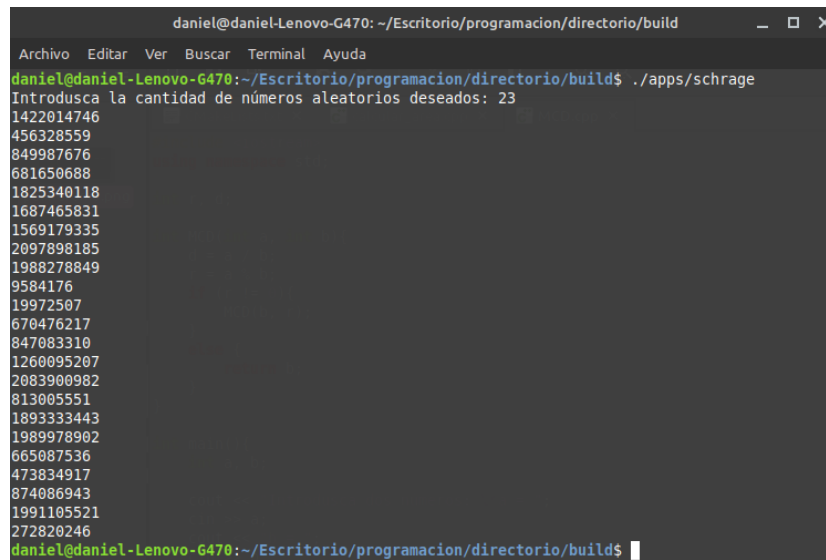
4.2. Algoritmo.

Para solucionar el problema computacional, seguimos las instrucciones del algoritmo de Schrage [5]. Escogiendo $I_0 = 1234567$ como el valor de la semilla

El código fuente está disponible en mi repositorio de git hub. [4]

4.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: $n = 23$. La salida del programa se observa en la Figura 4.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/schrage
Introduzca la cantidad de números aleatorios deseados: 23
1422014746
456328559
849987676
681650688
1825340118
1687465831
1569179335
2097898185
1988278849
9584176
19972507
670476217
847083310
1260095207
2083900982
813005551
1893333443
1989978902
665087536
473834917
874086943
1991105521
272820246
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 4: Ejecución de algunas instancias del problema.

5. Conclusiones.

En algunas ocasiones la recursividad puede ser una herramienta practiva al programar algún algoritmo en vez de utilizar ciclos `for` o `while`, aunque sea menos intuitivo que los ciclos, es más practico y util en muchos problemas. La declaración de métodos o funciones tambien son una herramienta imprescindible al programar ya que al reutilizar bloques enteros de códigos en distintas partes de nuestro programa nos ahorra tiempo y varias lineas de código.

Referencias

- [1] Daniel Reyes Barrera. *Problema 1-Algoritmo*. 2020. URL: https://github.com/danield877/cpp2020/blob/master/Clase_5_Tareas/calcular_area.cpp (visitado 24-11-2020).
- [2] Daniel Reyes Barrera. *Problema 2-Algoritmo*. 2020. URL: https://github.com/danield877/cpp2020/blob/master/Clase_5_Tareas/MCD.cpp (visitado 24-11-2020).

- [3] Daniel Reyes Barrera. *Problema 3-Algoritmo*. 2020. URL: https://github.com/danield877/cpp2020/blob/master/Clase_5_Tareas/palindromo.cpp (visitado 24-11-2020).
- [4] Daniel Reyes Barrera. *Problema 4-Algoritmo*. 2020. URL: https://github.com/danield877/cpp2020/blob/master/Clase_5_Tareas/schrage.cpp (visitado 24-11-2020).
- [5] *Generador de numeros aleatorios*. 2020. URL: http://catarina.udlap.mx/u_dl_a/tales/documentos/lfa/blanca_r_a/apendiceD.pdf (visitado 24-11-2020).
- [6] wikipedia.org. *Algoritmo de Euclides*. 2020. URL: https://es.wikipedia.org/wiki/Algoritmo_de_Euclides#:~:text=El%20algoritmo%20de%20Euclides%20es,divisor%20como%20una%20combinaci%C3%B3n%20lineal. (visitado 24-11-2020).