Conceptos Básicos

M.C. Carlos Rojas Sánchez[1]

[1]Licenciatura en Informática
Universidad del Mar :: Campus Puerto Escondido

.: Diseño Estructurado de Algoritmos :.

Saludos

Bienvenidos

Presentaciones

yo

¿Por qué de la asignatura?

ustedes

¿Nombre?

¿Qué espero del curso?[1]

[1] no se vale decir “pasar el curso”

Objetivo

Ser capaz de plantear metodológicamente la solución de problemas a través del manejo de técnicas de diseño estructurado y formulación de algoritmos.

¿Qué es el diseño estructurado de algoritmos?

¿Qué es el diseño estructurado de algoritmos?

- Modularidad
  - Dividir

¿Qué es el diseño estructurado de algoritmos?

- Modularidad
  - Dividir
- Descomposición por refinamientos sucesivos

¿Qué es el diseño estructurado de algoritmos?

- Modularidad
  - Dividir
- Descomposición por refinamientos sucesivos
- Módulos Independientes

Datos del curso

- Temario
- Evaluación
- Bibliografía

Para empezar

María prepara una cena especial para su familia. Ha hecho 56 canapés de queso y 40 de paté. Quiere repartir los canapés en el máximo número de platos posibles, de manera que haya el mismo número de canapés de cada tipo en todos los platos. ¿Cuántos platos necesitará?

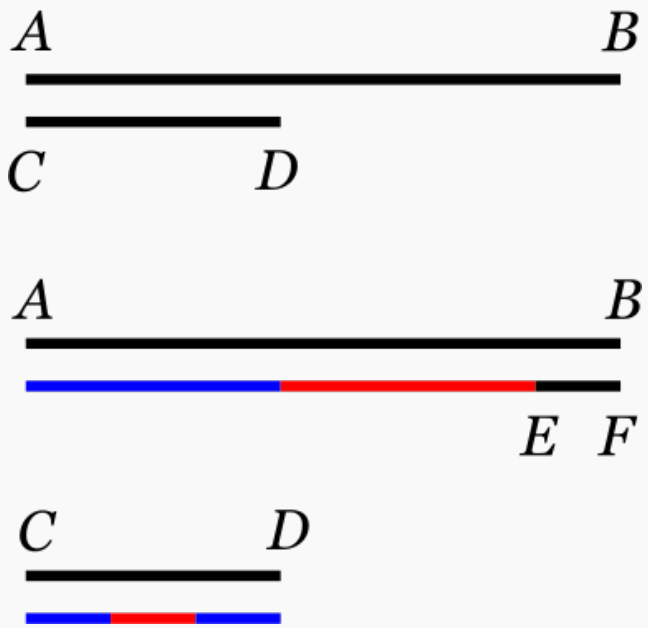
Solución

El número de platos deberá ser divisor de 56 y de 40. Además deberá ser el máximo divisor común, por lo que el problema se resuelve haciendo el mcd (56, 40)

$$mcd\,(56, 40) = \quad 2^3 = \quad 8$$

María necesitará 8 platos.

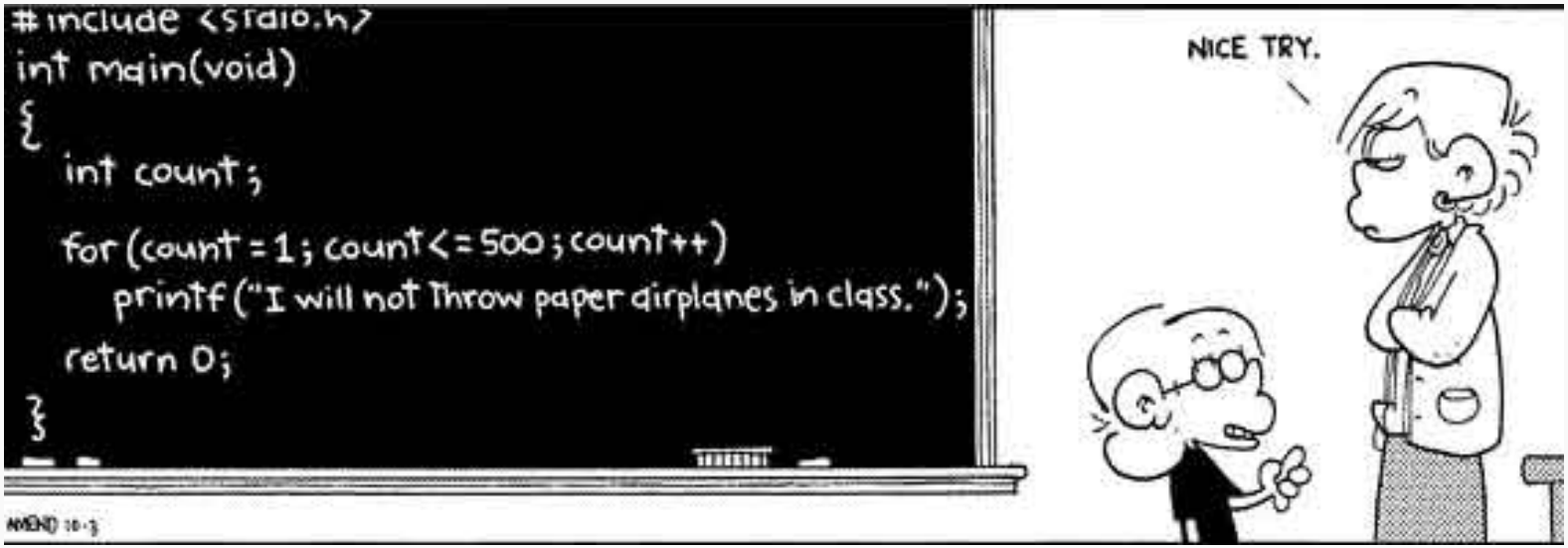
Forma gráfica



Forma escrita

- Dados dos segmentos AB y CD (con AB>CD), restamos CD de AB tantas veces como sea posible. Si no hay residuo, entonces CD es la máxima medida común.
- Si se obtiene un residuo EA, éste es menor que CD y podemos repetir el proceso: restamos EA tantas veces como sea posible de CD. Si al final no queda un residuo, EA es la medida común. En caso contrario obtenemos un nuevo residuo FC menor a EA.
- El proceso se repite hasta que en algún momento no se obtiene residuo. Entonces el último residuo obtenido es la mayor medida común.

Otros casos


#include <stdio.h>
int main(void)
{
int count;
for (count = 1; count <= 500; count++)
printf ("I will not throw paper airplanes in class.");
return 0;
}
NICE TRY.

1 Definición de lenguaje de programación

Lenguaje de Programación

Lenguaje de Programación

- Es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

Lenguaje de Programación

- Es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras.
- Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

2 Lenguajes de programación

Programación, ¿?

Lenguajes de programación

[2] Poner video “Hola mundo”

Lenguajes de programación

- Es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.[2]

[2] Poner video "Hola mundo"

Ahora si, la tarea

- Estudiar
  - SW para Pseudocódigo
    - PSeInt (http://pseint.sourceforge.net/)
  - SW para código en C
    - Dev-C++

Mantras de cada día :)

- Programar es una tarea fácil
- La práctica fortalece la habilidad de programar
- La programación es útil para todas las áreas del conocimiento

3 Algoritmos

Al-Juarismi

Abu Abdallah Muḥammad ibn Mūsā al-Jwārizmī (Abu Yāffar), conocido generalmente como al-Juarismi, fue un matemático, astrónomo y geógrafo persa musulmán, que vivió aproximadamente entre 780 y 850.

4 Definición de algoritmo

Algoritmo

- Conjunto de pasos ordenados y finitos que permiten resolver un problema o tarea específica.
- Los algoritmos son independientes de los lenguajes de programación y de la computadora que se vaya a emplear para ejecutarlo.

Características de un algoritmo

Finito En tamaño o número de instrucciones (tiene un primer y un último paso) y tiempo de ejecución (debe terminar en algun momento). Por lo tanto, debe tener un punto particular de inicio y fin.

Características de un algoritmo

Finito En tamaño o número de instrucciones (tiene un primer y un último paso) y tiempo de ejecución (debe terminar en algun momento). Por lo tanto, debe tener un punto particular de inicio y fin.

Preciso Debe tener un orden entre los pasos.

Características de un algoritmo

Finito En tamaño o número de instrucciones (tiene un primer y un último paso) y tiempo de ejecución (debe terminar en algun momento). Por lo tanto, debe tener un punto particular de inicio y fin.

Preciso Debe tener un orden entre los pasos.

Definido No debe ser ambiguo (dobles interpretaciones); si se ejecuta el mismo algoritmo el resultado siempre será el mismo, sin importar las entradas proporcionadas.

Características de un algoritmo

Finito	En tamaño o número de instrucciones (tiene un primer y un último paso) y tiempo de ejecución (debe terminar en algun momento). Por lo tanto, debe tener un punto particular de inicio y fin.
Preciso	Debe tener un orden entre los pasos.
Definido	No debe ser ambiguo (dobles interpretaciones); si se ejecuta el mismo algoritmo el resultado siempre será el mismo, sin importar las entradas proporcionadas.
General	Debe tolerar cambios que se puedan presentar en la definición del problema.

5 Representación de algoritmos

Representación de algoritmos

- Gráfica - Diagrama de flujo
- No Gráfica - Pseudocódigo

Texto narrativo

No gráfico

Consiste en escribir paso a paso las acciones o procedimientos que se deben realizar para resolver un problema. Para describir cada una de las acciones se utiliza lenguaje natural. Esta técnica es bastante sencilla, pero no es muy recomendable ya que puede llegar a ser muy ambiguo.

Pseudocódigo

No gráfico

Al igual que el texto narrativo, esta técnica describe paso a paso las acciones o procedimientos, pero utilizando un lenguaje de alto nivel, compacto e informal. Se utilizan las convenciones estructurales de un lenguaje de programación, pero está pensado para que una persona pueda entenderlo. Una de sus principales ventajas es que es independiente del lenguaje de programación, pero el principal problema es que no existe una sintaxis estándar para su uso.

Diagramas de flujo

Consiste en una representación gráfica de un algoritmo mediante símbolos bien definidos, que representan los pasos de un algoritmo, y el flujo es representado mediante flechas que conectan los puntos de inicio y fin del proceso.

6 Algoritmos cotidianos

Algoritmos cotidianos

- Se refiere a todos aquellos algoritmos que nos ayudan a resolver problemas diarios, y que los hacemos casi sin darnos cuenta de que estamos siguiendo una metodología para resolverlos

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.
  - Quitar la llanta.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.
  - Quitar la llanta.
  - Poner la llanta de repuesto.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.
  - Quitar la llanta.
  - Poner la llanta de repuesto.
  - Poner los tornillos.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.
  - Quitar la llanta.
  - Poner la llanta de repuesto.
  - Poner los tornillos.
  - Apretar los tornillos.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.
  - Quitar la llanta.
  - Poner la llanta de repuesto.
  - Poner los tornillos.
  - Apretar los tornillos.
  - Bajar el gato.

Ejemplo 1

Algoritmos cotidianos

- Diseñar un algoritmo para cambiar una llanta a un coche
  - Inicio.
  - Traer gato.
  - Levantar el coche con el gato.
  - Aflojar tornillos de las llantas.
  - Sacar los tornillos de las llantas.
  - Quitar la llanta.
  - Poner la llanta de repuesto.
  - Poner los tornillos.
  - Apretar los tornillos.
  - Bajar el gato.
  - Fin

Ejemplo 2

Algoritmos cotidianos

- Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido, en caso contrario rechazar el pedido.
  - Inicio

Ejemplo 2

Algoritmos cotidianos

- Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido, en caso contrario rechazar el pedido.
  - Inicio
  - Leer el pedido

Ejemplo 2

Algoritmos cotidianos

- Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido, en caso contrario rechazar el pedido.
  - Inicio
  - Leer el pedido
  - Examinar ficha del cliente

Ejemplo 2

Algoritmos cotidianos

- Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido, en caso contrario rechazar el pedido.
  - Inicio
  - Leer el pedido
  - Examinar ficha del cliente
  - Si el cliente es solvente aceptar pedido, en caso contrario rechazar pedido

Ejemplo 2

Algoritmos cotidianos

- Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido, en caso contrario rechazar el pedido.
  - Inicio
  - Leer el pedido
  - Examinar ficha del cliente
  - Si el cliente es solvente aceptar pedido, en caso contrario rechazar pedido
  - Fin

Ejemplo 3

Algoritmos cotidianos

- Determinar el mayor de tres números enteros.
  - Comparar el primero y el segundo entero, deduciendo cuál es el mayor.

Ejemplo 3

Algoritmos cotidianos

- Determinar el mayor de tres números enteros.
  - Comparar el primero y el segundo entero, deduciendo cuál es el mayor.
  - Comparar el mayor anterior con el tercero y deducir cuál es el mayor. Este será el resultado.

Ejercicio

- Proponer el algoritmo del ejemplo 3

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.
  - Obtener el segundo número (entrada), denominado NUM2.

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.
  - Obtener el segundo número (entrada), denominado NUM2.
  - Compara NUM1 con NUM2 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar NUM1. Llamar a este número MAYOR.

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.
  - Obtener el segundo número (entrada), denominado NUM2.
  - Compara NUM1 con NUM2 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar NUM1. Llamar a este número MAYOR.
  - Obtener el tercer número (entrada), y se denomina NUM3.

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.
  - Obtener el segundo número (entrada), denominado NUM2.
  - Compara NUM1 con NUM2 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar NUM1. Llamar a este número MAYOR.
  - Obtener el tercer número (entrada), y se denomina NUM3.
  - Compara MAYOR con NUM3 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar el MAYOR. Denominar a este número MAYOR.

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.
  - Obtener el segundo número (entrada), denominado NUM2.
  - Compara NUM1 con NUM2 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar NUM1. Llamar a este número MAYOR.
  - Obtener el tercer número (entrada), y se denomina NUM3.
  - Compara MAYOR con NUM3 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar el MAYOR. Denominar a este número MAYOR.
  - Presentar el valor MAYOR (salida).

Ejercicio

- Proponer el algoritmo del ejemplo 3
  - Inicio
  - Obtener el primer número (entrada), denominado NUM1.
  - Obtener el segundo número (entrada), denominado NUM2.
  - Compara NUM1 con NUM2 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar NUM1. Llamar a este número MAYOR.
  - Obtener el tercer número (entrada), y se denomina NUM3.
  - Compara MAYOR con NUM3 y seleccionar el mayor ; si los dos enteros son iguales, seleccionar el MAYOR. Denominar a este número MAYOR.
  - Presentar el valor MAYOR (salida).
  - Fin

Bibliografía I

- Joyanes Aguilar, Luis.
  Fundamentos de programación, algoritmos y estructura de datos.
  McGraw-Hill. 2003, 3ª Edición.
- Cairó Battistutti, Osvaldo Gabriel.
  Metodología de la programación: algoritmos, diagramas de flujo y programas.
  Alfaomega. 2005, 3ª Edición.