Tarea 1. Programacion y Algoritmos

Carlos Giovanny Encinia Gonzalez

August 2021

Problema 1

Resumen de la siguiente publicación: S. Kumar and P. K. Singh, "An overview of modern cache memory and performance analysis of replacement policies," 2016 IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 210-214, doi: 10.1109/ICETECH.2016.7569243.

Respuesta

El cache es un tipo de memoria semiconductora que permite acceder y almacenar subconjuntos de datos o instrucciones que se utilizan de manera frecuente sin tener que recorrer lugares de almacenamiento mas lejanos como la memoria principal.

El rendimiento de la memoria caché es calculado sobre la base de miss rate, miss penalty y el average access time. el miss rate se define como la fracción de memoria que no este en el caché mientras que el hit rate es la fracción de acceso a la memoria que se encuentra en la caché. Miss penalty esta definido como el numero total de ciclos que la CPU se detiene durante el acceso a la memoria determinado por la suma de Ciclos (tiempo) para reemplazar un bloque en la caché. Las tres unidades principales de un procesador son Instruction, Execution, y Storage Unit. La Instruction se encarga de organizar las instrucciones de un programa y las codifica y decodifica. La unidad Execution realiza operaciones lógicas, aritméticas. el Storage Unit establece interfaz a través de un almacenamiento temporal entre otras dos unidades. Los componentes esenciales de la unidad de almacenamiento son memoria caché, translator, y Translation Look-aside Buffer. Para disminuir la transferencia de datos a la memoria cache, esta se encuentra ubicada en el procesador. El algoritmo de reemplazo juega un papel importante en el diseño de la memoria de chache porque toma la decisión de seleccionar una línea particular en la memoria y debe ser reemplazada con la memoria principal deseada. Los procesadores modernos emplean politicas de cache de reemplazo como LRU, Random, FIFO, LFU. LRU es política de sustitución de caché mejor escalable. La velocidad de los procesadores crece continuamente tal es que la latencia de memoria disminuye, por lo que eliminar las pérdidas de caché es muy importante para el rendimiento general del procesador.

Problema 2

Describir detalladamente los paradigmas de computación principales y su relación. Dar ejemplos de cada paradigma.

Respuesta:

Programacion imperativa

Este paradigma esta relacionado con la creacion de hardware, mas bien dicho la logica que hace funcionar al hardware, esta pensada para poder ejecutar de manera ordenada una serie de instrucciones un ejemplo claro de este paradigma lo podemos encontrar en el lenguaje ensamblador.

Programacion imperativa

```c
int i = 0;
i = 35 + i;
i = i/45;
printf("%d\n", i);
```

Programacion estructurada

En este paradigma se emplean los controles de ciclo, los condicionales, y se separa en codigo en bloques, todo esto para tener una buena ingenieria de software.

Programacion estructurada

```
int main()
{
    int i = 0;

    while(i < 0)
    {
        if(i=2)
        {
        printf("es 2");
        }
        i++;
    }

    return 0;
}
```

Programación procedimental

Se caracteriza porque el programa se divide en subtareas o funciones que se pueden llamar de manera sucesiva, se presentan diferentes tipos e variables como las locales y las globales. Ejemplos de lenguajes en donde se puede usar este paradigma son, C, C++, python, Java.

Programacion procedimental

```
int suma(int a, int b)
{
return a+b;
}

int main()
{
    int a, b;
    a = suma(a, b);
    return 0;
}
```

Programación orientada a objetos

En lugar de usar funciones, este paradigma utiliza objetos y etodos, esto es una especia de estructura de datos que describe de manera abstracta algo, esto con el fin de ahorrar tiempo, es muy utilizada en el rubro de los videojuegos, en este paradigma existen los conceptosde herencia, encapsulamiento y polimorfismo.

Programacion orientada a objetos

```
class Triangulo: public Fig
{
 public:
  Triangulo(float lado)
  {
    l = lado;
  }
  float perimetro()
  {
    return 4 * l;
  }
  float area()
  {
    return l * l;
  }
};
```

Programación funcional

La primitiva en este paradigma de programacion es la funcion, y la funcion se toma como la definicion de funcion matematica, la importancia de este paradigma llace en que se trata todo lo posible de alejarse de la programacion maquina y se utilizan funciones de manera elegante. Ejemplo de lenguajes que utilizan este paradigma son Elixir, Scala.

Programacion funcional

```
defmodule funcion do
  def print(opcion) do
    opcion
    |> get_message
    |> IO.puts
  end

  defp get_message(respuesta) do
    cond  do
      respuesta == 1 -> "Hola"
      respuesta == 2 -> "Hola 2"
      respuesta -> "Adios"
    end
  end
end
```

Programación declarativa

En este paradigma siempre se describe cual es el re-

sultado final deseado, en otras palabras se describe lo que se debe de hacer pero no como hacerlo. Existen dos tipos de programacion declarativa, la programacion logica y la programacion funcional.

Programación logica

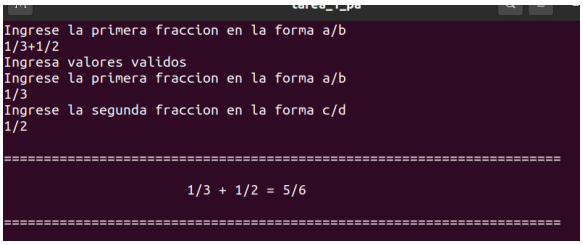
Este paradigma utiliza la logica de predicados para poder funcionar. Un ejemplo de lenguale de programacion que utiliza este paradigma es Prolog.

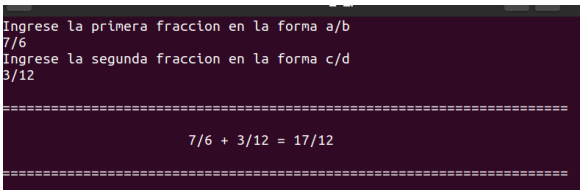
Programacion logica

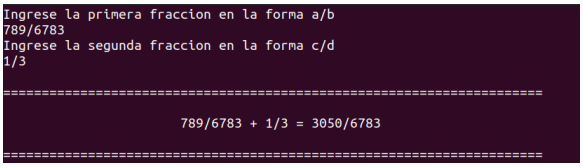
```
?- X is 1+5.
   X = 6

?- X = 1+5.
   X = 1+5

?- 1+5 =:= 3+3.
   yes
```



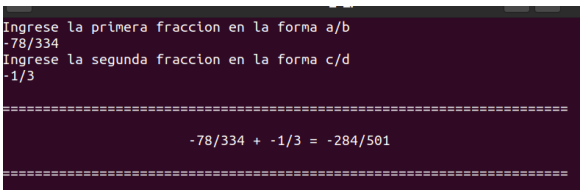


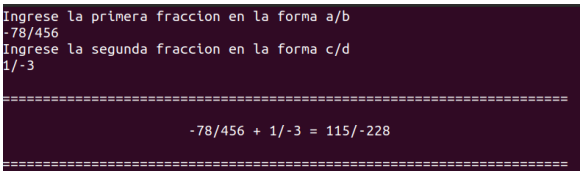


Problema 3

Programa que sume 2 fracciones, la entrada debe ser:

Introducir Fracción 1: a/b
Introducir Fracción 2: c/d
La salida debe ser: a/b + c/d = f/g
donde a-f son números enteros.
Nota: Hay que leer sobre entrada de datos con símbolos especiales de la función scanf().





Respuesta

Se creo un programa que lee dos fracciones de la forma $\frac{a}{b} + \frac{c}{d}$ y da como resultado $\frac{e}{f}$, lee los datos como strings y hace la respectiva conversion a long, esto para que se puedan realizar grandes sumas de fracciones y el resultado se trata de simplificar utilizando el mcd. En seguida se muestran algunos resultados obtenidos. Despues se muetra el codigo que se implemento. Es importante mencionar que la funcion scanf() no puede leer los espacios o saltos de lineas por lo que al ingresar estos valores, el programa falla.

Code 1: Problema 3 suma de fracciones

```c
//Giovanny Encinia
//17 agosto 2021
//suma de fracciones

#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <ctype.h>
#include <string.h>
#define ZERO  0
#define ONE 1
#define TAMANIO 24
#define TAMANIO_2 12
#define TEN 10


int mcd(long a, long b);
int separador_input(char  *nombre, char *forma, int termino);

long a, b, c, d; // variable global, se almacenan los inputs



int mcd(long a, long b)
{
    /*Encuentra el maximo comun divisor de dos numeros*/
    long temp;

    if(b == ZERO)
    {
        return a;
    }

    else
    {
        if(b > a)
        {
            temp = a;
            a = b;
            b = temp;
        }
        return mcd(b, a % b);
    }
}

int separador_input(char  *nombre, char *forma, int termino)
{
    /*Esta funcion separa el numerador y el denominador del input
```

```
tomando en cuenta que existe el simbolo /
si hay mas de este simbolo o algun caracter que no
sea un entero, pide al usuario que ingrese de nuevo
los datos

Parametros
==========
char *nombre: hace referencia el termino de la suma, primer o segundo
cahr *forma: pide datos a/b o c/d
int termino: un flag para saber donde guardar el resultado

Return
======
int ciclo: retorna 1 o 0, si el caracter no esta permitido retorna 1
*/
int count_slash = ZERO, ciclo = ONE;
int index_sep;
int aprobado_1 = ONE, aprobado_2 = ONE;
char x[TAMANIO], x_1[TAMANIO_2], x_2[TAMANIO_2];
int i = ZERO, j = ZERO, k = ZERO;

count_slash = ZERO;
printf("Ingrese la %s fraccion en la forma %s\n", nombre, forma);
scanf("%s", x);


while(x[i] != '\0') //recorre la cadena
{

    if(x[i] == '/') //  busca el / para separar denom y num
    {
        count_slash++; // cuenta cuantos slash hay
        index_sep = i; // guarda el indice del /
    }

    i++;
}

while(j < index_sep)  //se guarda el numerador
{
    x_1[j] = x[j];

    if(!isdigit(x_1[j]))
    {

        if(j== 0 && x_1[j] == '-') //asegura negativos - al inicio
        {
            aprobado_1 = ONE;
        }
        else
```

```
            {
                aprobado_1 = ZERO;
            }

        }

        j++;
    }

    x_1[j] = '\0';
    j++;

    while(x[j] != '\0') // lo que queda del original
    {

        x_2[k] = x[j]; // se guarda el denominador

        if(!isdigit(x_2[k]))
        {
            //asegura negativos -
            if((j == index_sep + 1) && (x_2[k] == '-'))
            {
                aprobado_2 = ONE;
            }
            else
            {
                aprobado_2 =ZERO;
            }

        }

        j++;
        k++;

    }
    x_2[k] = '\0';

    if(count_slash != ONE || \
       aprobado_1 == ZERO || aprobado_2 == ZERO)//comprobacion
    {
        ciclo = ONE;
        printf("Ingresa valores validos\n");
    }
    else
    {
        ciclo = ZERO;

        if(termino == ONE)
        {
            a = (long) strtol(x_1, NULL, TEN);
```

```
            b = (long) strtol(x_2, NULL, TEN);
        }
        else
        {
            c = (long) strtol(x_1, NULL, TEN);
            d = (long) strtol(x_2, NULL, TEN);
        }

    }
    return   ciclo;
}

int main()
{
    int ciclo = ONE;
    long numerador, denominador, divisor;

    while(ciclo)
    { // mientras haya inconsistencias ciclo
        ciclo = separador_input("primera", "a/b", ONE);

        if(ciclo == ONE)
        {
            continue;
        }

        ciclo += separador_input("segunda", "c/d", 2);
    }

    numerador = a * d + b * c;
    denominador = b * d;
    // lo usaremos para simplificar
    divisor = mcd(fabs(numerador), fabs(denominador));
    numerador /= divisor;
    denominador /= divisor;

    printf("\n=============================================\
==========================\n");
    printf("\n                                \
%ld/%ld + %ld/%ld = \%ld/%ld\n\n", \
           a, b, c, d, numerador, denominador);
    printf("=============================================\
==========================\n");

    return 0;
}
```