5 de noviembre del 2021

Práctica 1

Análisis y Diseño de Algoritmos

MIGUEL ANGEL BALTAZAR COYOTL
ESCUELA SUPERIOR DE CÓMPUTO

Introducción

La multiplicación es una operación básica que conocemos desde la educación básica, sin embargo en ocasiones al ser muy largos los números a multiplicar se suele complicar al hacerlo a mano o incluso si se hace por un programa, por lo cual con la estrategia divide y vencerás, la cual como su nombre lo dice al dividir los números que se multiplicaran y aplicar una serie de pasos se logra conseguir, así mismo usando esta estrategia junto a la multiplicación de Gauss de consigue obtener una operación que es la siguiente:

X * Y = 2^n (XI)(YI) + 2^n/2((XI+XD) (YD+YI) – XIYI -YDXD) + (XD) (YD)

Con esta operación es que se consigue que el algoritmo implementado en el código que está a continuación funcione.

Código en C++

La parte de divide y venceras y multiplicacion de Gauss esta en la página 10

```
#include <iostream>
#include <ctime>
#include <fstream>
#include <string.h>
#include <cmath>
#include <sstream>
using namespace std;

int bindec(string bina);
long double bindec1(string bina);
int espotdos(int);
string convpotdos(int sizelon, int size, string bin);
void recibebinario(string &,string &);
int longitud(string &str1, string &str2);
```

```
string suma(string first, string second);
string decimal_binstr(int numero);
string resta(string lhs, string rhs);
string multiplica(string X, string Y);
void recibedecimal(string &bin1,string &bin2);
int hex_dec();
void recibehexadecimal(string &bin1,string &bin2);
string multpot(string str, int stepnum);
string hexadecimal(string num);
string binarioHex(string binario);

int main(){
	string bin1,bin2;
	int lon;
	int base,resul;
	cout<<"Ingresa la longitud del numero con mayor cantidad de digitos\n";
	cin>>lon;
	cout<<"Ingresa la base: \n";
	cout<<"2 para binario \n";
	cout<<"10 para decimal \n";
	cout<<"16 para hexadecimal \n";
	cin>>base;
	switch(base){
		case 2:
			recibebinario(bin1,bin2);
			cout<<multiplica(bin1,bin2);
			break;
		case 10:
			recibedecimal(bin1,bin2);
```

```
                printf("%.0Lf",bindec1(multiplica(bin1,bin2)));
                break;
        case 16:
                recibehexadecimal(bin1,bin2);
                cout<<binarioHex(multiplica(bin1,bin2));
                break;
        default: cout<<"No esta en las bases";
    }

    return 0;
}

long double bindec1(string bina){
    int potdos,lon;
    long double decimal = 0;
    int n,suma;
    lon = bina.size();
    n = lon-1;
    for(int i=0;i<lon;i++){
        suma = bina[i];
        decimal += (pow(2,n) * (suma-48));
        n--;
    }
    return decimal;

}


int bindec(string bina){
```

```
        int decimal = 0;
        int lon,n,suma;
        lon = bina.size();
        n = lon-1;
        for(int i=0;i<lon;i++){
                suma = bina[i];
                decimal += (pow(2,n) * (suma-48));
                n--;
        }
        return decimal;
}

int espotdos(int lon){
        int npotdos = 1;
        if((lon & (lon - 1)) == 0){
                return lon;
        }
        else{
                while (npotdos < lon)
    npotdos <<= 1;
        return npotdos;
        }
}

string convpotdos(int sizelon, int size, string bin){
        if(((sizelon & (sizelon - 1)) == 0) && (sizelon == size)){
                return bin;
        }
        else{
```

```
                string sub;
                for(int i = 0; i < (size - sizelon); i++){
                        sub.append("0");
                }
                sub.append(bin);
                return sub;
        }
}

void recibebinario(string &bin1,string &bin2){
        int lonbin1,lonbin2,longi;
        cout<<"Ingrese el primer numero ";
        cin>>bin1;
        cout<<"\nIngrese el segundo numero ";
        cin>>bin2;
        lonbin1 = bin1.size();
        lonbin2 = bin2.size();
        longi = espotdos(max(lonbin1,lonbin2));
        bin1 = convpotdos(lonbin1,longi,bin1);
        bin2 = convpotdos(lonbin2,longi,bin2);
}


int longitud(string &str1, string &str2){
        int len1 = str1.size();
        int len2 = str2.size();
        if (len1 < len2){
                for (int i = 0 ; i < len2 - len1 ; i++){
                        str1 = '0' + str1;
```

```
                }
                return len2;
        }
        else if (len1 > len2){
                for (int i = 0 ; i < len1 - len2 ; i++){
                        str2 = '0' + str2;
                }
        }
        return len1;
}


string suma(string first, string second){
        string result;
        int length = longitud(first, second);
        int carry = 0;

        for (int i = length-1 ; i >= 0 ; i--){
                int firstBit = first.at(i) - '0';
                int secondBit = second.at(i) - '0';
                int sum = (firstBit ^ secondBit ^ carry)+'0';
                result = (char)sum + result;
                carry = (firstBit&secondBit) | (secondBit&carry) | (firstBit&carry);
        }
        if (carry){
                result = '1' + result;
        }

        return result;
```

```
}

string decimal_binstr(int numero){
   string binario = "";
   if (numero > 0) {
      while (numero > 0) {
         if (numero%2 == 0) {
            binario = "0"+binario;
         } else {
            binario = "1"+binario;
         }
         numero = (int) numero/2;
      }
   } else if (numero == 0) {
      binario = "0";
          }
          return binario;
}

string resta(string lhs, string rhs){

          int length = longitud(lhs, rhs);
          int diff;
          string result;

          for (int i = length-1; i >= 0; i--){
                    diff = (lhs[i]-'0') - (rhs[i]-'0');
                    if (diff >= 0){
                              result = decimal_binstr(diff) + result;
```

```
                }
                else{
                        for (int j = i-1; j>=0; j--){
                                lhs[j] = ((lhs[j]-'0') - 1) % 10 + '0';
                                if (lhs[j] != '1'){
                                        break;
                                }
                        }
                        result = decimal_binstr(diff+2) + result;
                }
        }
        return result;
}

string multpot(string str, int stepnum){
        string shifted = str;
        for (int i = 0 ; i < stepnum ; i++){
                shifted = shifted + '0';
        }
        return shifted;
}
```

```
// Divide y venceras junto a multiplicacion de Gauss
string multiplica(string x, string y){
        int n;
        string Xi,Xd,Yi,Yd,P1,P2,P3;
        n = longitud(X, Y);

        if (n == 1) return ((Y[0]-'0' == 1) && (X[0]-'0' == 1)) ? "1" : "0";

        int mit = n/2;
        int mit2 = (n-mit);
        //Divide y venceras
        Xi = x.substr(0, mit);
        Xd = x.substr(mit, mit2);
        Yi = y.substr(0, mit);
        Yd = y.substr(mit, mit2);

        P1 = multiplica(Xi, Yi);
        P2 = multiplica(suma(Xi, Xd), suma(Yi, Yd));
        P3 = multiplica(Xd, Yd);
        //Multiplicacion de Gauss 2^np1 + base^(n/2)*p2-p1-p3 + p3
        return suma(suma(multpot(P1, 2*(n-n/2)),P3),multpot(resta(P2,suma(P1,P3)), n-(n/2)));
}
```

```
void recibedecimal(string &bin1,string &bin2){
        int lonbin1,lonbin2,num1,num2,longi;
        cout<<"Ingrese el primer numero ";
        cin>>num1;
        cout<<"\nIngrese el segundo numero ";
        cin>>num2;
        bin1 = decimal_binstr(num1);
        bin2 = decimal_binstr(num2);
        lonbin1 = bin1.size();
        lonbin2 = bin2.size();
        longi = espotdos(max(lonbin1,lonbin2));
        bin1 = convpotdos(lonbin1,longi,bin1);
        bin2 = convpotdos(lonbin2,longi,bin2);
}

int hex_dec(){
        char  NroHexa[100];
   char Aux[2];
   int  i, Error;
   int NroDec;
   long PosMult;

   gets( NroHexa );
   for( Error = 0, i = strlen(NroHexa) - 1; i>=0; i--)
      {
      if( !( ( NroHexa[i] >= 'A' && NroHexa[i] <= 'F' ) ||
           ( NroHexa[i] >= 'a' && NroHexa[i] <= 'f' ) ||
           ( NroHexa[i] >= '0' && NroHexa[i] <= '9' ) ) )
         {
```

```
        Error = 1;
        break;
        }
    }

if( !Error )
    {
    for( NroDec = 0, i = strlen(NroHexa) - 1; i>=0; i--)
        {
        PosMult = (long)pow( 16, (strlen(NroHexa) - 1 - i) );
        if( PosMult == 0 )
            PosMult = 1;
        switch( NroHexa[i] )
            {
            case 'A':
            case 'a':
                NroDec += 10 * PosMult;
                break;
            case 'B':
            case 'b':
                NroDec += 11 * PosMult;
                break;
            case 'C':
            case 'c':
                NroDec += 12 * PosMult;
                break;
            case 'D':
            case 'd':
                NroDec += 13 * PosMult;
```

```
                break;
            case 'E':
            case 'e':
                NroDec += 14 * PosMult;
                break;
            case 'F':
            case 'f':
                NroDec += 15 * PosMult;
                break;
            default:
                Aux[0] = NroHexa[i];
                Aux[1] = '\0';
                NroDec += atoi( Aux ) * PosMult;
                break;
            }
          }
      }
      return NroDec;
}
void recibehexadecimal(string &bin1,string &bin2){
        int lonbin1,lonbin2,num1,num2,longi;
        cout<<"Ingrese el primer numero ";
        fflush(stdin);
        num1 = hex_dec();
        cout<<"\nIngrese el segundo numero ";
        num2 = hex_dec();
        bin1 = decimal_binstr(num1);
        bin2 = decimal_binstr(num2);
        lonbin1 = bin1.size();
```

```
            lonbin2 = bin2.size();
            longi = espotdos(max(lonbin1,lonbin2));
            bin1 = convpotdos(lonbin1,longi,bin1);
            bin2 = convpotdos(lonbin2,longi,bin2);
}
string hexadecimal(string num){
   if(num=="0000"){
      return "0";
   }else if(num=="0001"){
      return "1";
   }else if(num=="0010"){
      return "2";
   }else if(num=="0011"){
      return "3";
   }else if(num=="0100"){
      return "4";
   }else if(num=="0101"){
      return "5";
   }else if(num=="0110"){
      return "6";
   }else if(num=="0111"){
      return "7";
   }else if(num=="1000"){
      return "8";
   }else if(num=="1001"){
      return "9";
   }else if(num=="1010"){
      return "A";
   }else if(num=="1011"){
```

```
        return "B";
    }else if(num=="1100"){
        return "C";
    }else if(num=="1101"){
        return "D";
    }else if(num=="1110"){
        return "E";
    }else if(num=="1111"){
        return "F";
    }
      return 0;
}
string binarioHex(string binario)
{
    string b = binario+"";
    string sol = "";//solucion
    int numeroSeparaciones = binario.length()/4;
    if(binario.length()%4>0){//si el numero no es exacto añade una separacion
        numeroSeparaciones++;
    }
    for (int var = binario.length(); var < numeroSeparaciones*4; ++var) {//si el numero no es esacto añade 0 a la derecha
        b="0"+b;
    }
    for (int var = 0; var < numeroSeparaciones; ++var) {//coge los digitos binarios de 4 en 4 y llamando al metodo anterior, los pasa a hexadecimal
        sol+=hexadecimal(b.substr((var*4),4));
    }
    return sol;
}
```

Pruebas especificadas

Binario:

```
Ingresa la longitud del numero con mayor cantidad de digitos
1
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
2
Ingrese el primer numero 1

Ingrese el segundo numero 0
0
--------------------------------
```

```
Ingresa la longitud del numero con mayor cantidad de digitos
5
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
2
Ingrese el primer numero 11010

Ingrese el segundo numero 1010
000000100000100
```

```
Ingresa la longitud del numero con mayor cantidad de digitos
8
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
2
Ingrese el primer numero 10110101

Ingrese el segundo numero 101
000001110001001
--------------------------------
```

Decimal:

```
Ingresa la longitud del numero con mayor cantidad de digitos
1
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
10
Ingrese el primer numero 8

Ingrese el segundo numero 9
72
--------------------------------
```

```
Ingresa la longitud del numero con mayor cantidad de digitos
5
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
10
Ingrese el primer numero 12345

Ingrese el segundo numero 34
419730
--------------------------------
```

```
Ingresa la longitud del numero con mayor cantidad de digitos
7
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
10
Ingrese el primer numero 1234567

Ingrese el segundo numero 1234
1523455678
--------------------------------
```

Hexadecimal:

```
Ingresa la longitud del numero con mayor cantidad de digitos
1
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
16
Ingrese el primer numero A

Ingrese el segundo numero 4
28
--------------------------------
```

```
Ingresa la longitud del numero con mayor cantidad de digitos
5
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
16
Ingrese el primer numero 123AB

Ingrese el segundo numero D3
0000000000F065F1
--------------------------------
```

```
Ingresa la longitud del numero con mayor cantidad de digitos
7
Ingresa la base:
2 para binario
10 para decimal
16 para hexadecimal
16
Ingrese el primer numero 111222E

Ingrese el segundo numero 12CD
000000140F1F9AD6
--------------------------------
```