int Tong(int n)

{

if (n == 1)

return 1;

else

return n + Tong(n - 1);

}

int Binhphuongtong(int n)

{

if (n == 1)

return 1;

else

return n\*n + Binhphuongtong(n - 1);

}

double Tongmotphanle(int n)

{

if (n == 1 || n == 0)

return 1;

else

return 1.0/n + Tongmotphanle(n - 2.0);

}

int giaithua(int n)

{

if (n == 0)

return 1;

else

return n\*giaithua(n - 1);

}

double tonggiaithua(int n)

{

if (n == 1)

return 1;

else

return giaithua(n)+tonggiaithua(n - 1);

}

**Viết hàm đệ qui tính giá trị của x n ; với x, n là số nguyên. Sau đó xây dựng chương trình để mô phỏng và kiểm tra hàm vừa xây dựng.**

double XmuN(int x, int n)

{

if (n == 0 )

return 1;

if (n > 0) return 1.0\*XmuN(x, n - 1)\*x;

return XmuN(x, n + 1)/x;

}

**Viết hàm đệ qui đếm số chữ số của một số nguyên không âm**

int Count( int n)

{

static int ctr = 0;

if (n != 0)

{

ctr++;

Count(n / 10);

}

return ctr;

}

**Viết hàm đệ qui trả về số đảo ngược của một số nguyên không âm.**

int SoDaoNguoc(int n)

{

static int kq = 0;

if (n>0)

{

kq = kq \* 10 + n % 10;

SoDaoNguoc(n / 10);

}

return kq;

}

**Viết hàm đệ qui tính tổng các chữ số chẵn/lẻ của một số nguyên không âm.**

int Sumchan(int n)

{

if (n == 0)

return 0;

if ((n % 10) % 2 == 0)

return n % 10 + Sumchan(n / 10);

return Sumchan(n / 10);

}

int Sumle(int n)

{

if (n == 0)

return 0;

if ((n % 10) % 2 != 0)

return n % 10 + Sumle(n / 10);

return Sumle(n / 10);

}

**Viết hàm đệ qui chuyển một số nguyên dương từ hệ thập phân sang nhị phân**

int Chuyen(int n)

{

if (n == 0)

return 0;

return n % 2 + Chuyen(n / 2) \* 10;

}

**Viết hàm đệ qui chuyển một số nguyên ở dạng nhị phân sang hệ thập phân.**

int Chuyen(int n, int mu)

{

if (n == 0)

return 0;

return (n % 10) \* pow(2, mu) + Chuyen(n / 10, mu+1);

}

* **Cho mu =0 ở hàm main**

**Viết hàm đệ qui tìm ước số chung lớn nhất của hai số nguyên**

int UCLN(int a, int b)

{

if (b == 0)

return a;

return UCLN(b, a%b);

}

**Viết hàm đệ qui tính tổng các giá trị của một mảng một chiều lưu trữ số nguyên.**

int Sum(int arr[MAX], int n)

{

if (n == 0) return 0;

return arr[n - 1] + Sum(arr, n - 1);

}

**Viết hàm đệ qui tìm giá trị nhỏ nhất của một mảng một chiều lưu trữ số nguyên.**

int Minarr(int arr[MAX], int n)

{

if (n == 0) return 0;

static int min = INT\_MAX;

if (min > arr[n - 1])

min = arr[n - 1];

Minarr(arr, n - 1);

return min;

}

int Maxarr(int arr[MAX], int n)

{

if (n == 0) return 0;

static int max = INT\_MIN;

if (max < arr[n - 1])

max = arr[n - 1];

Maxarr(arr, n - 1);

return max;

}

**Viết hàm đệ qui để xuất ra một tam giác Pascal**

int pascal(int r, int c)

{

if (c == 1) return 1;

if (c == r) return 1;

return pascal (r - 1, c)+ pascal(r - 1, c - 1);

}

void Output(int n)

{

for (int i = 1; i <= n; i++)

{

for (int j = 1; j <= i; j++)

cout << setw(4) << pascal(i, j);

cout << endl;

}

}