

BÀI GIẢNG KỸ THUẬT LẬP TRÌNH



NGUYỄN ĐÌNH CƯỜNG
Khoa CNTT Đại học Nha Trang

LẬP TRÌNH TÍNH TOÁN SỐ LỚN

- Xử lý số nguyên

Bài toán tràn số

Đổi chỗ hai số nguyên : a và b

a = 3 b=5

- Cộng và nhân hai số lớn a và b

A= 111111111122222222222222222222222233333333333333336666666666666666

[illegible]

Xử lý số lớn với C#, Java : 2^{64}

- Lập trình máy tính chứng minh bài toán Fermat

$$a^n + b^n = c^n$$

- Tính $N!$



```

• 1 // Printing multiple lines with a single statement
• 2 #include <iostream>
• 3
• 4 // function main begins program execution
• 5 int main()
• 6 {
• 7     std::cout << "Welcome\nto\nnC++!\n";
• 8
• 9     return 0; // indicate that program ended successfully
• 10
• 11 } // end function main

```

```

• 1 // Addition program.
• 2 #include <iostream>
• 4 // function main begins program execution
• 5 int main()
• 6 {
• 7     int integer1; // first number to be input by user
• 8     int integer2; // second number to be input by user
• 9     int sum;      // variable in which sum will be store
• 11 std::cout << "Enter first integer\n"; // prompt
• 12 std::cin >> integer1;                // read an integer
• 14 std::cout << "Enter second integer\n"; // prompt
• 15 std::cin >> integer2;                // read an integer
• 17 sum = integer1 + integer2; // assign result to sum
• 19 std::cout << "Sum is " << sum << endl; // print sum
• 21 return 0; // indicate that program ended successfully
• 23 } // end function main

```

- Các toán tử trên bit chỉ có tác dụng trên các kiểu số nguyên:

& And
 | Or
 ^ Xor
 ~ Not
 << Shift Left
 >> Shift Right

Số mã bù 2 = số bù 1 + 1

```
char a, b, c;
```

```
a=-5;
```

```
b=3;
```

Số bù 2 của a: a=-5: 0000 0101
 1111 1010 + 1
 1111 1011
 b=3 0000 0011

```
c = a & b    0000 0011
```

```
c = a & b;
```

```
printf ("\n %d",c);
```

Ví dụ chứng minh tính chất bắc cầu

- $A \rightarrow B$ $B \rightarrow C$ thì $A \rightarrow C$

Viết chương trình máy tính

Kiểu con trỏ

- Các biến trước đây đều là biến có kích thước và kiểu dữ liệu xác định. Gọi các biến kiểu này là biến tĩnh. Khi khai báo biến tĩnh, các ô nhớ sẽ được cấp phát mà không biết trong quá trình chạy chương trình có sử dụng hết chúng hay không.
- Các biến tĩnh dạng này sẽ tồn tại trong suốt thời gian thực thi chương trình dù có những biến mà chương trình chỉ sử dụng 1 lần rồi bỏ. Các hạn chế về biến tĩnh:
 - Cấp phát ô nhớ dư, gây ra lãng phí ô nhớ.
 - Cấp phát ô nhớ thiếu, chương trình thực thi bị lỗi.
- Để giải quyết những hạn chế trên, ngôn ngữ C cung cấp cho ta một loại biến đặc biệt gọi là biến động với các đặc điểm sau:
 - Chỉ phát sinh trong quá trình chạy chương trình chứ không phát sinh lúc bắt đầu chương trình.
 - Khi chạy chương trình, kích thước của biến, vùng nhớ và địa chỉ vùng nhớ được cấp phát cho biến có thể thay đổi.
 - Sau khi sử dụng xong có thể giải phóng để tiết kiệm chỗ trong bộ nhớ.
- Vì thế, ngôn ngữ C lại cung cấp cho ta một loại biến đặc biệt nữa để khắc phục tình trạng này, đó là biến con trỏ (pointer) với các đặc điểm:
 - Biến con trỏ không chứa dữ liệu mà chỉ chứa địa chỉ của dữ liệu hay chứa địa chỉ của ô nhớ
 - Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu, luôn có kích thước cố định là 2 bytes nếu trong cùng 1 đoạn và 4 bytes nếu khác đoạn.

Kiểu con trỏ(tiếp)

a. Khai báo biến con trỏ

Cú pháp: <Kiểu> * <biến con trỏ>

2 byte quản lý
biến con trỏ

Ví dụ 1: Khai báo 2 biến a,b có kiểu int và 2 biến pa, pb là 2 biến con trỏ kiểu int.

```
int a, b, *pa, *pb;
```

Ví dụ 2: Khai báo biến f kiểu float và biến pf là con trỏ float

```
float f, *pf;
```

Address	Value	Comment
0013FFC4	7C81604F	RETURN to kernel32.7C81604F
0013FFC8	7C918738	ntdll.7C918738
0013FFD0	7FFD7400	
0013FFD4	0054AD70	
0013FFD8	0013FFC8	
0013FFDC	85677400	
0013FFE0	FFFFFFFF	End of SEH chain
0013FFE4	7C8399F3	SE handler
0013FFE8	7C816058	kernel32.7C816058
0013FFFC	00000000	
0013FFF0	00000000	
0013FFF4	00000000	
0013FFF8	00000000	

```
int a, *pa;
```

```
a=5;
```

```
pa=&a;
```

```
a=5
```

```
0x12F'F:0x003F'
```

trỏ tới vùng nhớ

5

```
*pa=5;
```

Nội dung vùng nhớ
trỏ tới *pa=5

```
0x00FF:0x0011
```

Địa chỉ con trỏ &pa

Cộng, trừ con trỏ với một số nguyên: Có thể cộng (+), trừ (-) 1 con trỏ với 1 số nguyên N nào đó; kết quả trả về là 1 con trỏ. Con trỏ này chỉ đến vùng nhớ cách vùng nhớ của con trỏ hiện tại N phần tử.

- Đơn vị tăng hay giảm của con trỏ có kích thước của biến được trỏ đến

Con trỏ được dùng như mảng

Ví dụ: Cho 1 mảng 1 chiều các số nguyên a có 5 phần tử, truy cập các phần tử theo kiểu mảng và theo kiểu con trỏ.

```
#include <stdio.h>
#include <conio.h>
/* Nhập mảng bình thường*/
voidNhapMang(int a[], int N)
{
    int i;
    for (i=0;i<N;i++)
    {
        printf("Phan tu thu %d: ",i);
        scanf("%d",&a[i]);
    }
}
/* Nhập mảng theo dạng con trỏ*/
voidNhapConTro(int a[], int N)
{
    int i;
    for(i=0;i<N;i++)
    {
        printf("Phan tu thu %d: ",i);
        scanf("%d",a+i);
    }
}

int main()
{
    int a[20],N,i;
    printf("So phan tu N= ");scanf("%d",&N);
    NhapMang(a,N);
    printf("Truy cap theo kieu mang: ");
    for(i=0;i<N;i++) printf("%d ",a[i]);
    printf("\nTruy cap theo kieu con tro: ");
    for(i=0;i<N;i++) printf("%d ",*(a+i));
    getch();
    return 0;
}
```

Con trỏ được dùng như mảng

- `a` tương đương với `&a[0]`
- `a + i` tương đương với `&a[i]`
- `*(a+i)` tương đương với `a[i]`

Ví dụ:

```
float a[30], *p;
p=a;
```

Bốn cách viết sau có tác dụng như nhau:

```
a[i]    *(a+i)    *(p+i)    p[i]
```

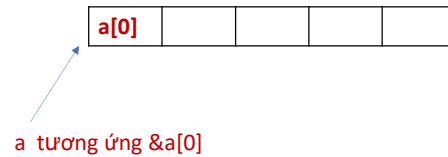
Đối là mảng một chiều

```
int *pa;
float *pa;
double *pa;
```

Có thể khai báo như mảng hình thức

```
int pa[];
float pa[];
double pa[];
```

Truy cập đến phần tử `a[i]`, ta sử dụng: `*(a+i)` và `pa[i]`



Con trỏ được dùng như mảng 2 chiều

- `&a[i][j]` : lấy địa chỉ là không hợp lệ trong một số IDE C/C++

```
for (i=0; i<n;i++)
    for (j=0; j<n;j++)
    {
        printf("\n a[%d][%d]=",i,j);
        scanf("%d",&a[i][j]);
    }

printf("\n");
for (i=0; i<n;i++)
{
    for (j=0; j<n;j++)
        printf("%d%c",a[i][j],32);
    printf("\n");
}
```

Chú ý

Cấp phát vùng nhớ cho con trỏ và sử dụng như mảng 1 chiều

Cấp phát vùng nhớ cho con trỏ và sử dụng như mảng 2 chiều

Truyền tham số trong mảng

- Sử dụng từ khóa `const` không làm thay đổi giá trị trong thân hàm

```
void nhan_mt(const int a[20][20], const int b[20][20], int c[20][20], int n)
{
    int i, j, k;
    for (i=1 ; i<=n; ++i)
        for (j=1; j<=n; ++j)
        {
            c[i][j]=0;
            for (k=1; k<=n; ++k)
                c[i][j]+=a[i][k]*b[k][j];
        }
}
```

a. Hàm cấp phát:

`void *malloc(size_t size)`: Cấp phát vùng nhớ có kích thước là size.

`void *calloc(size_t nitems, size_t size)`: Cấp phát vùng nhớ có kích thước là nitems*size.

Ví dụ: Giả sử ta có khai báo:

```
int a, *pa, *pb; pa = (int*)malloc(sizeof(int)); /* Cấp phát vùng nhớ có kích thước bằng với
                                                kích thước của một số nguyên */
```

```
pb= (int*)calloc(10, sizeof(int)); /* Cấp phát vùng nhớ có thể chứa được 10 số nguyên*/
```

Lưu ý: Khi sử dụng hàm `malloc()` hay `calloc()`, ta phải ép kiểu vì nguyên mẫu các hàm này trả về con trỏ kiểu `void`.

b. Thu hồi vùng nhớ: Một vùng nhớ đã cấp phát cho biến động do biến con trỏ giữ địa chỉ, khi không còn sử dụng nữa, ta sẽ thu hồi lại vùng nhớ này nhờ hàm `free()`.

Cú pháp: `void free(void *block)`

Ý nghĩa: Giải phóng vùng nhớ được quản lý bởi con trỏ `block`.

Ví dụ: Ở ví dụ trên, sau khi thực hiện xong, thu hồi vùng nhớ cho 2 biến con trỏ `pa` và `pb` như sau:

```
free(pa); free(pb);
```

Con trỏ hàm

- Khai báo con trỏ hàm và mảng con trỏ hàm

```

        (*f) (float)
        (*mf[5])(int)
double (*g) (int, double)
        (*mg[30])(double, float)

#include <stdio.h>
double fmax(double x, double y)
{
    return (x>y?x:y);
}

double (*pf) (double, double) = fmax;
int main ()
{
    printf("\n max=%f", pf(4.5, 78.9));
    return 1;
}

```

Con trỏ hàm

- Dùng mảng con trỏ để lập bảng giá trị

```

        y=x2  y=sin(x)  y=cos(x)  exp(x)  sqrt(x)

#include "stdio.h"
#include "math.h"

double bp(double x)
{
    return x*x;
}

int main()
{
    int j, double x=1.0;
    // con trỏ hàm
    typedef double (*ham)(double);
    ham f[6];    // hoặc có thể khai báo như sau: double
    (*f[6])(double);

    f[1]=bp; f[2]=sin; f[3]=cos; f[4]=exp; f[5]=sqrt;

    while (x<=10.0)
    {
        printf ("\n");
        for (j=1; j<=5; ++j)
            printf("%10.2f", f[j][x]);
        x+=0.5;
    }
}

```

BỘ NHỚ CHƯƠNG TRÌNH

- Bộ nhớ chương trình được chia làm 4 phần

Vùng mã lệnh (chứa mã lệnh và hằng)

Vùng cấp phát tĩnh (chứa đối tượng ngoài và tĩnh)

Vùng cấp phát động (Heap)

Vùng ngăn xếp (chứa các đối tượng cục bộ)

Địa chỉ cao



Vùng nhớ tự do sử dụng chung giữa Heap và Stack

BỘ NHỚ CHƯƠNG TRÌNH

Dùng con trỏ hàm trở tới hàm khởi động lại của DOS (restart), đặt tại địa chỉ 0xFFFF:0x0000

```

/*Chương trình kiểm tra mật
khẩu*/
#include <stdio.h>
#include<conio.h>
#include<dos.h>
#include<ctype.h>

char mk[]="ABC";
void far (*f) (void) ;
void main ()
{
    char ch; int i;
    int dung=1;
    clrscr();
    printf("\n Vào mat khau 3
    ki tu:");
    i=0;
    while (i<3)
    {
        ch=getch();
        if (toupper(ch)!=mk[i])
        {
            dung =0;
            break;
        }
        ++i;
    }
    if (!dung)
    {
        printf("\n Sai mat khau"); getch();
        f=MK_FP(0xFFFF,0x0000);
        f();
    }
    else
    {
        printf("\n Dung mat khau");
        getch();
    }
}

```


GỌI NGẮT CỦA DOS

▪ Gọi ngắt của DOS

```
union REGS v, r;
v.h.ah=5; // chức năng số 5
v.h.al=t; // số hiệu trang màn hình cần hiển thị
int86(0x10, &v, &r); // Thực hiện ngắt 0x10
```

▪ Chuyển đổi địa chỉ

```
unsigned FP_SEG(địa_chỉ_thực)
unsigned FP_OFF(địa_chỉ_thực)
```

Chuyển địa chỉ phân đoạn sang địa chỉ thật

```
void far *MK_FP(seg, off)
```

Ví dụ: `char buf[100];`
`unsigned ds, dx;`
`ds = FP_SEG(buf); dx = FP_OFF(buf);`

Ta có câu lệnh truy cập địa chỉ xa

```
char far *pchar;
pchar = (char far *)MK_FP(0x800:0);
```

Cần 4 byte quản lý
biến con trỏ far

GỌI NGẮT CỦA DOS

▪ Cấu trúc khai báo ngắt

```
struct WORDREGS /*Các thanh ghi chung 16 bit*/
{
    unsigned int ax, bx, cx, dx, si, di, cflag, fflag;
};

struct BYTEREGS /*Các thanh ghi chung 8 bit*/
{
    unsigned int al, ah, bl, bh, cl, ch, dl, dh;
};

struct SREGS
{
    unsigned int es, cs, ss, ds;
};

union REGS /*Hợp giữa thanh ghi 16bit và thanh ghi 8 bit*/
{
    struct WORDREGS w;
    struct BYTEREGS b;
}
```

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>
void main()
{
    unsigned char far *p;
    int n;
    int k;
    unsigned seg, off;
    p=(unsigned char far *) MK_FP(0,0);
    clrscr();
    while (1)
    {
        printf("\n So hieu ngat bam 0 ket thuc:");
        scanf("%d", &n);
        if (n==0) break;
        k= (n-1)*4;
        off=p[k]+256*p[k+1];      seg=p[k+2] +256*p[k+3];
        printf("\n Dia chi: %x:%x",seg, off);
    }
}
```

```
#include<dos.h>
#include<conio.h>
#include<stdlib.h>

#define VT 132 // vị trí thông báo
char far *p_mh= (char *) MK_FP(0xB800, 0);
unsigned long far *t_time = (unsigned long far *) MK_FP(0, 0x46C);

char buf_time[] = {'T', 47, 'I', 47, 'M', 47, 'E', 47, ':', 47, 32, 47, 32, 47, 32, 47, 32, 47, 32, 47, 32, 47, 32, 47, 32};
char buf_luu[28];
void thong_bao_thoi_gian()
{
    // luu trang thai man hinh
    for (int i=0; i<28; ++i)
        buff_luu[i] = p_mh[i];
    // xác định giờ, phút, giây
    int gio = (int) (*t_time/65543);
    unsigned long du = *t_time%65543;
```

```

int phut = (int ) (du/1092) ;
du = du % 1092;

int giay = (int) (du/18);

buff_time[12] = gio /10 +48;
buff_time[14] = gio %10 +48;

buff_time[18] = phut /10 +48;
buff_time[20] = phut%10 +48;

buff_time[24] = giay /10 +48;
buff_time[26] = giay%10 +48;

// đưa thông báo ra màn hình
for (i=0; i<28; ++i)
    p_mh[i] =buf_time [i];
getch();

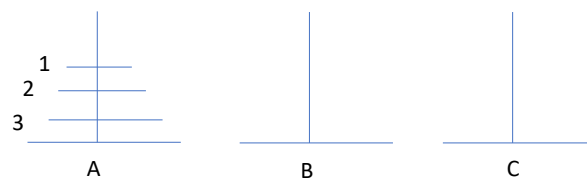
// khôi phục màn hình
for (i=0; i<28; ++i) p_mh [i]=
buf_luu[i];
}

void main()
{
    int ch1, ch2;
    clrscr();
    while (1)
    {
        if (kbhit())
        {
            ch1 = getch ();
            if (ch 1 == 0)  ch2=getch();
            if (ch1==27) break;
            if (ch1 ==0 && ch2==59)
                thông_bao_thoi_gian();
        }
        // in ra chữ cái ngẫu nhiên
        gotoxy(random(80)+1, random(25)+1);
        putchar (random(26)+65);
        delay(400);
    }
}

```

KỸ THUẬT ĐỆ QUY VÀ QUY HOẠCH ĐỘNG

Bài toán: Tháp Hà Nội



Nếu $m = 1$, chuyển vị trí đầu đến vị trí cuối.

- Chuyển ($m-1$, vị trí đầu, vị trí cuối, vị trí giữa)
- Chuyển (1, vị trí đầu, vị trí giữa, vị trí cuối)
- Chuyển ($m-1$, vị trí giữa, vị trí đầu, vị trí cuối)

$N=3$

Chuyển (N , 'A', 'B', 'C');

Bài toán: Tháp Hà Nội

```
#include<stdio.h>
#include<conio.h>

void chuyen(int m, char dau, char giua, char cuoi)
{
    if (m==1) printf("\n chuyen %c ->%c", dau, cuoi);
    else
    {
        chuyen(m-1, dau, cuoi, giua );
        chuyen(1, dau, giua, cuoi);
        chuyen(m-1, giua, dau, cuoi);
    }
}

int main()
{
    int n;
    n=3;
    chuyen(n, 'A', 'B', 'C');
    getch();
    return 1;
}
```

```
"C:\Users\ngdic\OneDrive\Desktop\bai giang\thap_ha..."
chuyen A ->C
chuyen A ->B
chuyen C ->B
chuyen A ->C
chuyen B ->A
chuyen B ->C
chuyen A ->C
Process returned 0 (0x0)   execution time : 3.373 s
Press any key to continue.
```

Thuật toán sinh

```
void generate()
{
    <xây dựng cấu hình ban đầu>;
    stop=false;
    while not stop do
    {
        <Đưa ra cấu hình đang có>;
        Sinh_kế_tiếp;
    }
}
```

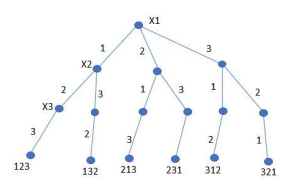
Ví dụ

In ra các hoán vị của số từ 1..N

Thuật toán quay lui

```
void try(int i)
{
    int j;
    for ( j=1; j < ni ; j++ )
        if ( <chấp nhận j> )
        {
            < Xác nhận xi theo j >;
            if (i==n ) <ghi nhận một cấu hình>;
            else
                try (i+1);
        }
}

int main()
{
    // ...các khai báo dữ liệu
    try(1);
    return 1;
}
```



```

1  #include<stdio.h>
2  #include<conio.h>
3
4  int a[10], b[10];
5  int n;
6
7  void out_result()
8  {
9      int i;
10     for( i=1; i<=n; i++) printf("%d", a[i]);
11     printf("\n");
12 }
13
14 void sinh(int i)
15 {
16     int j;
17     for (int j=1; j<=n; j++)
18         if (b[j]==0)
19         {
20             a[i]=j;
21             b[j]=1;
22             if (i==n) out_result();
23             else
24                 sinh(i+1);
25             b[j]=0;
26         }
27 }
28
29 int main()
30 {
31     n=4;
32     sinh(1);
33     getch();
34     return 1;
35 }
36

```

Đánh dấu phần từ đã sử dụng

Sinh phần từ kế tiếp

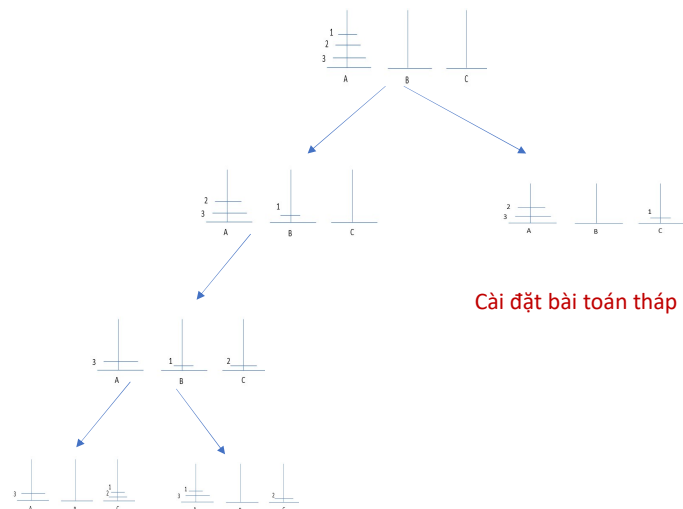
```

"C:\Users\ngdic\OneDrive\Desktop\bai giang\hoan_v...
1243
1324
1342
1423
1432
2134
2143
2314
2341
2413
2431
3124
3142
3214
3241
3412
3421
4123
4132
4213
4231
4312
4321

Process returned 0 (0x0)   execution time : 3.126 s
Press any key to continue.

```

Bài toán: Tháp Hà Nội Quy Quay Lui



Bài toán Phân tích số N thành tổng các số nguyên

Nhận xét tổng bằng S, giá trị số sau lớn hơn hoặc bằng giá trị số trước

N= 5

S= 1 + 1 + 1 + 1 + 1

S= 1 + 1 + 1 + 2

S= 1 + 1 + 3

S= 1 + 2 + 2

S= 1 + 4

S= 2 + 3

N-gram speech
recognition

```
#include<stdio.h>
#include<conio.h>

int S, M;
int a[10];
int n=10;

void output_result()
{
    int i;
    i=1;
    while ((a[i]!=0)) {printf("%d\\c", a[i], 32); i=i+1;}
    printf("\\n");
}

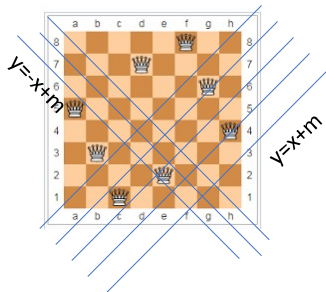
int tong(int i)
{
    int j;
    for (j=a[i-1]; j<=M; j++)
        if (S+j<=M)
        {
            S=S + j;
            a[i]=j;
            if (S==M) output_result();
        }
    else
        tong(i+1);
    S=S-j;
    a[i]=0;
}
```

Giá trị mời

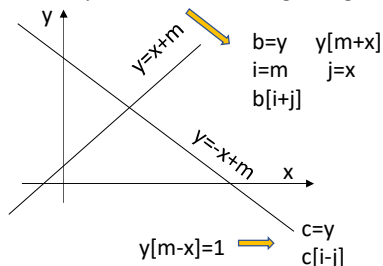
```
int main()
{
    S=0; M=8;
    for(int i=0; i<n; i++) a[i]=0; a[0]=1;
    tong(1);
    getch();
    return 1;
}
```

```
"C:\Users\ngduc\OneDrive\Desktop\bai giang\tong.exe"
1 1 1 1 1 1 1 1
1 1 1 1 1 1 2
1 1 1 1 1 3
1 1 1 1 2 2
1 1 1 1 4
1 1 1 2 3
1 1 1 5
1 1 2 2 2
1 1 2 4
1 1 3 3
1 1 6
1 2 2 3
1 2 5
1 3 4
1 7
2 2 2 2
2 2 4
2 3 3
2 6
3 5
4 4
8
Process returned 0 (0x0)   execution time : 5.898 s
Press any key to continue.
```

Bài toán 8 Quân Hậu Trên Bàn Cờ



$y[x+m]=1$ khi đường thẳng bị kiểm soát



Thuật toán quay lui

```
void try(int i)
{
    int j;
    for (j=1; j < n; j++)
        if ( a[j] && b[i+j] && c[i-j] )
        {
            x[i]=j;      // ghi nhận trạng thái mới
            a[j] = 1;
            b[i+j]=1;
            c[i-j]=1;
            if (i==n ) output_result();
            else
                try (i+1);
            // trả lại trạng thái cũ
            a[j]=0;
            b[i+j]=0;
            c[i-j]=0;
        }
}
```

Bài toán 8 Quân Hậu Trên Bàn Cờ

```

#include<stdio.h>
#include<conio.h>
int a[10], b[20], c[20];
int x[10];
int n=8; //so vua
void output_result()
{
    int i;
    for (i=1; i<=n; i++) printf("%d%c",x[i],32);
    printf("\n");
}

void quaylui(int i)
{
    int j;
    for ( j=1; j <= n ; j++ )
        if ( (a[j]==0) && (b[i+j]==0) && (c[i-j+n]==0) )
        {
            x[i]=j; //cot
            a[j] = 1; //hang
            b[i+j]=1;
            c[i-j+n]=1;
            if (i==n) output_result();
            else
                quaylui(i+1);
            x[i]=0;
            a[j]=0;
            b[i+j]=0;
            c[i-j+n]=0;
        }
}

```

```

int main()
{
    int i;

    //clear();
    for (i=1; i<=n; i++) {a[i]=0; c[i]=0;x[i]=0;}
    for (i=1; i<=2*n; i++) b[i]=0;
    quaylui(1);
    getch();
    return 1;
}

```

```

"C:\Users\ngduc\OneDrive\Desktop\bai giang\quanhau.exe"
1 5 8 6 3 7 2 4
1 6 8 3 7 4 2 5
1 7 4 6 8 2 5 3
1 7 5 8 2 4 6 3
2 4 6 8 3 1 7 5
2 5 7 1 3 8 6 4
2 5 7 4 1 8 6 3
2 6 1 7 4 8 3 5
2 6 8 3 1 4 7 5
2 7 3 6 8 5 1 4
2 7 5 8 1 4 6 3
2 8 6 1 3 5 7 4
3 1 7 5 8 2 4 6
3 5 2 8 1 7 4 6
3 5 2 8 6 4 7 1
3 5 7 1 4 2 8 6
3 5 8 4 1 7 2 6
3 6 2 5 8 1 7 4
3 6 2 7 1 4 8 5
3 6 2 7 5 1 8 4
3 6 4 1 8 5 7 2
3 6 4 2 8 5 7 1
3 6 8 1 4 7 5 2
3 6 8 1 5 7 2 4
3 6 8 2 4 1 7 5
3 7 2 8 5 1 4 6
3 7 2 8 6 4 1 5
3 8 4 7 1 6 2 5
4 1 5 8 2 7 3 6

```

BÀI TOÁN QUY HOẠCH ĐỘNG

Bài toán 1 Chia kẹo

Có N gói kẹo S_1, S_2, \dots, S_n Mỗi gói có K_1, K_2, \dots, K_n . Hãy chia số kẹo thành 2 phần sao cho độ lệch 2 phần là nhỏ nhất

S_1	S_2	S_3	S_4	S_5	S_6	S_7
10	8	4	5	7	12	6

Thành lập tất cả các tổng có thể , tổng nào gần với tổng số kẹo chia 2 thì lấy

BÀI TOÁN QUY HOẠCH ĐỘNG

Đối sánh 2 chuỗi A và B để đạt được maximum score

Các thao tác xử lý

- Kí tự A[i] và B[j] trùng khớp **match** : không làm gì cả +2 score
- Kí tự A[i] và B[j] sai khác nhau **mismatch**, thay thế A[i] với B[j] : -1 score
- Chèn **insert** một khoảng trống vào A[i] : -1 score
- Xóa **delete** một kí tự từ A[i] : -1 score

Hướng dẫn cài đặt

2 Xâu A[1..n], B[1..m]

$V(i, j)$: tối ưu score A[1..i] và B[1..j]

$score(C_1, C_2)$: điểm kí tự C_1 đối sánh C_2

Ta có :

$V(0,0) = 0$;

$V(i,0) = i \times score(A[i],_)$ // delete A[1..i] $i > 0$

$V(0,j) = j \times score(_,B[j])$ // insert vào B[1..j] $j > 0$

Biểu thức hồi quy

Ví dụ:

Cho cặp chuỗi với xử lý bất kì

A='ACAATCC' -> 'A_CAATCC'

B='AGCATGC' -> 'AGCATGC_'

2-22 --2-

Alignment Score = $4 \times 2 + 4 \times (-1) = 4$

Với $i, j > 0$

$V(i, j) = \max(\text{option 1}, \text{option 2}, \text{option 3})$

option 1 = $V(i-1, j-1) + score(A[i], B[j])$: score of match and mismatch

option 2 = $V(i-1, j) + score(A[i], _)$: delete A_i

option 3 = $V(i, j-1) + score(_, B[j])$: insert B_j

STRING ALIGNMENT

A = 'xxx...xx' A = 'xxx...xx' A = 'xxx...x_'
 | | |
 B = 'yyy...yy' B = 'yyy...y_-' B = 'yyy...yy'
 match/mismatch delete insert

		_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7	
A	-1								
C	-2								
A	-3								
A	-4								
T	-5								
C	-6								
C	-7								

Base Cases

		_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7	
A	-1	2	1	0	-1	-2	-3	-4	
C	-2	1	1	3					
A	-3								
A	-4								
T	-5								
C	-6								
C	-7								

		_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7	
A	-1	2	-1	0	-1	-2	-3	-4	
C	-2	1	1	3	2	1	0	-1	
A	-3	0	0	2	5	4	3	2	
A	-4	-1	-1	1	4	4	3	2	
T	-5	-2	-2	0	3	6	5	4	
C	-6	-3	-3	0	2	5	5	7	
C	-7	-4	-4	-1	1	4	4	7	

Figure 6.1: Example: A = 'ACAATCC' and B = 'AGCATGC' (alignment score = 7)

Chuỗi tối ưu

A='A_CAAT[C]C'

B='AGC_AT[G]C'

Alignment score = $5 \times 2 + 3 \times (-1) = 7$ score

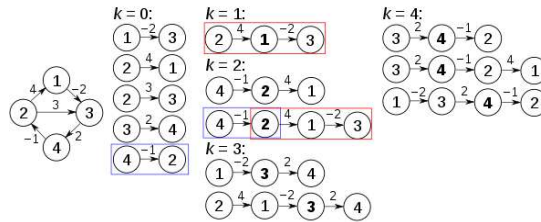
QUY HOẠCH ĐỘNG- Thuật toán Floyd-Warshall

```
for (int k=0; k<V; k++)
```

```
for (int i=0; i<V; i++)
```

```
for (int j=0; j<V; j++)
```

```
AdjMat[i][j] = min (AdjMat[i][j], AdjMat[i][k] + AdjMat[k][j]);
```



		j				
		1	2	3	4	5
i	1	1	0	∞	-2	∞
	2	4	0	3	∞	∞
	3	∞	∞	0	2	∞
	4	∞	-1	∞	0	∞

		j				
		1	2	3	4	5
i	1	1	0	∞	-2	∞
	2	4	0	2	∞	∞
	3	∞	∞	0	2	∞
	4	∞	-1	∞	0	∞

		j				
		1	2	3	4	5
i	1	1	0	∞	-2	∞
	2	4	0	2	∞	∞
	3	∞	∞	0	2	∞
	4	3	-1	1	0	∞

		j				
		1	2	3	4	5
i	1	1	0	∞	-2	0
	2	4	0	2	4	∞
	3	∞	∞	0	2	∞
	4	3	-1	1	0	∞

		j				
		1	2	3	4	5
i	1	1	0	-1	-2	0
	2	4	0	2	4	∞
	3	5	1	0	2	∞
	4	3	-1	1	0	∞

GỖ TIẾNG VIỆT UNICODE

```

xinchaocpp - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

xinchaocpp
1 // xinchaocpp : Defines the entry point for the console application.
2 //
3
4 #include "stdafx.h"
5
6 #include <iostream>
7 #include <iio.h>
8 #include <fcntl.h>
9 #include <string>
10 #include <windows.h>
11 #include <fstream>
12
13 int main()
14 {
15     _setmode(_fileno(stdin), _O_U16TEXT);
16     _setmode(_fileno(stdout), _O_U16TEXT);
17     SetConsoleTitle(L"Việt Nam Vô Địch!");
18     HANDLE hdlConsole = GetStdHandle(STD_OUTPUT_HANDLE);
19     CONSOLE_FONT_INFOEX consoleFont;
20     consoleFont.cbSize = sizeof(consoleFont);
21     GetCurrentConsoleFontEx(hdlConsole, FALSE, &consoleFont);
22     memcpy(consoleFont.FaceName, L"Consolas", sizeof(consoleFont.FaceName));
23     SetCurrentConsoleFontEx(hdlConsole, FALSE, &consoleFont);
24
25     std::wcout << L"Tiếng Việt có dấu" << std::endl;
26     std::wstring test;
27     std::wcout << L"Hãy nhập vào một chuỗi ký tự:" << std::endl;
28     std::getline(std::wcin, test);
29     std::wcout << L"Chuỗi ký tự mà bạn vừa mới nhập:" << std::endl;
30     std::wcout << test << std::endl;
31
32     FILE *f;
33 }
  
```

Việt Nam Vô Địch!

Tiếng Việt có dấu

Hãy nhập vào một chuỗi ký tự:

Xin chào các bạn sinh viên Việt Nam

Chuỗi ký tự mà bạn vừa mới nhập:

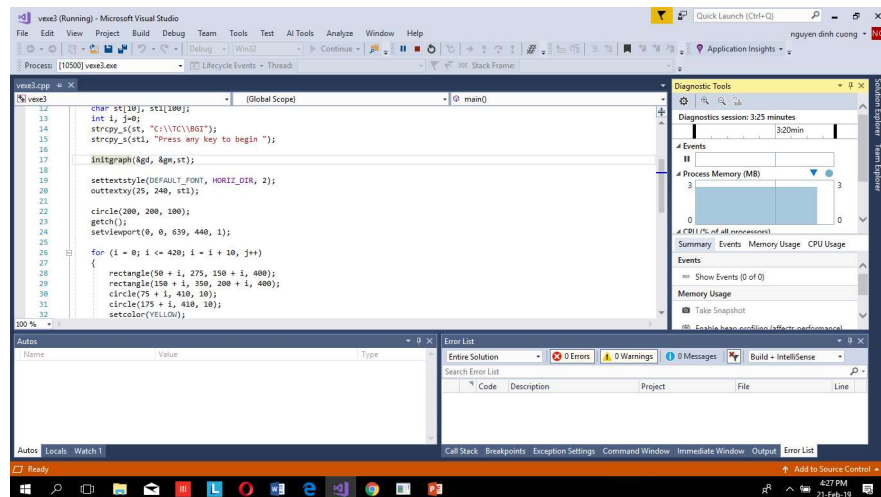
Xin chào các bạn sinh viên Việt Nam

Press any key to continue . . .

BẢNG MÃ UNICODE

00A0		ı	€	£	¤	¥	¦	§	¨	©	ª	«	¬	®	¯
00B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾
00C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î
00D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ
00E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î
00F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ
0100	Ā	ā	Ă	ă	Ą	ą	Ć	ć	Ĉ	ĉ	Č	č	Ď	ď	
0110	Đ	đ	Ē	ē	Ĕ	ĕ	Ė	ė	Ę	ę	Ė	ė	Ġ	ġ	
0120	Ģ	ģ	Ģ	ģ	Ĥ	ĥ	Ħ	ħ	İ	ı	İ	ı	İ	ı	
0130	Í	í	Ĳ	ĳ	Ĵ	ĵ	ķ	ķ	Ľ	ĺ	Ł	ł	Ł	ł	
0140	Į	į	Ļ	ļ	Ņ	ņ	Ņ	ņ	Ņ	ņ	Ņ	ņ	Ņ	ņ	
0150	Ō	ō	Œ	œ	Ř	ř	Ř	ř	Ř	ř	Ř	ř	Ř	ř	

THƯ VIỆN ĐỒ HỌA





STL LIBRARY

sort

```
#include <algorithm>
void sort( iterator start, iterator end );
void sort( iterator start, iterator end, StrictWeakOrdering cmp );

vector<int> v;
v.push_back( 23 );
v.push_back( -1 );
v.push_back( 9999 );
v.push_back( 0 );
v.push_back( 4 );
cout << "Before sorting: ";
for( unsigned int i = 0; i < v.size(); i++ )
{
    cout << v[i] << " ";
}
cout << endl;
sort( v.begin(), v.end() );
cout << "After sorting: ";
for( unsigned int i = 0; i < v.size(); i++ )
{
    cout << v[i] << " ";
}
cout << endl;
```

Before sorting: 23 -1 9999 0 4
After sorting: -1 0 4 23 9999

sort

```
int array[] = { 23, -1, 9999, 0, 4 };
unsigned int array_size = 5;
cout << "Before sorting: ";
for( unsigned int i = 0; i < array_size; i++ )
{
    cout << array[i] << " ";
}

cout << endl;
sort( array, array + array_size );
cout << "After sorting: ";
for( unsigned int i = 0; i < array_size; i++ ) {
    cout << array[i] << " ";
}
cout << endl;
```

sort

```

bool cmp( int a, int b ) {
    return a > b;
}

...
vector<int> v;
for( int i = 0; i < 10; i++ ) {
    v.push_back(i);
}
cout << "Before: ";
for( int i = 0; i < 10; i++ ) {
    cout << v[i] << " ";
}
cout << endl;
sort( v.begin(), v.end(), cmp );
cout << "After: ";
for( int i = 0; i < 10; i++ ) {
    cout << v[i] << " ";
}
cout << endl;

```

Vecto

```

#include <vector>
vector();
vector( const vector& c );
vector( size_type num, const TYPE& val = TYPE() );
vector( input_iterator start, input_iterator end );
~vector();

vector<int> v1( 5, 42 );

```

STACK

```
#include <stack>
bool empty() const;
```

```
stack<int> s;
for( int i = 0; i < 5; i++ ) {
    s.push(i);
}
while( !s.empty() ) {
    cout << s.top() << endl;
    s.pop();
}
```

Stack constructors	construct a new stack
empty	true if the stack has no elements
pop	removes the top element of a stack
push	adds an element to the top of the stack
size	returns the number of items in the stack
top	returns the top element of the stack

QUEUE

```
#include <queue>
bool empty() const;
```

```
for( int i = 0; i < 5; i++ ) {
    v.push_back(i);
}
while( !v.empty() ) {
    cout << v.back() << endl;
    v.pop_back();
}
```

```
queue<int> q;
for( int i=0; i < 10; i++ )
    q.push(i);
```

```
while( !s.empty() ) {
    cout << q.top() << " ";
    q.pop();
}
```

```
#include <queue>
queue();
queue( const Container& con );
```

```
queue<string> waiting_line;
while( waiting_line.size() < 5 ) {
    cout << "Welcome to the line, please enter your name: ";
    string s;
    getline( cin, s );
    waiting_line.push(s);
}
while( !waiting_line.empty() ) {
    cout << "Now serving: " << waiting_line.front() << endl;
    waiting_line.pop();
}
```

```
Welcome to the line, please enter your name: Nate
Welcome to the line, please enter your name: lizzy
Welcome to the line, please enter your name: Robert B. Parker
Welcome to the line, please enter your name: ralph
Welcome to the line, please enter your name: Matthew
Now serving: Nate
Now serving: lizzy
Now serving: Robert B. Parker
Now serving: ralph
Now serving: Matthew
```

LIST

```
#include <list>
bool empty() const;

vector<int> v;
for( int i = 0; i < 5; i++ ) {
    v.push_back(i);
}
while( !v.empty() ) {
    cout << v.back() << endl;
    v.pop_back();
}
```

LIST

```
// create a vector of random integers
cout << "original vector: ";
vector<int> v;
for( int i = 0; i < 10; i++ ) {
    int num = (int) rand() % 10;
    cout << num << " ";
    v.push_back( num );
}
cout << endl;
// find the first element of v that is even
vector<int>::iterator iter1 = v.begin();
while( iter1 != v.end() && *iter1 % 2 != 0 ) {
    iter1++;
}
// find the last element of v that is even
vector<int>::iterator iter2 = v.end();
do {
    iter2--;
} while( iter2 != v.begin() && *iter2 % 2 != 0 );
// only proceed if we find both numbers
if( iter1 != v.end() && iter2 != v.begin() ) {
    cout << "first even number: " << *iter1 << ", last even number: " << *iter2 << endl;
    cout << "new vector: ";
    vector<int> v2( iter1, iter2 );
    for( int i = 0; i < v2.size(); i++ ) {
        cout << v2[i] << " ";
    }
    cout << endl;
}
```

```
original vector: 1 9 7 9 2 7 2 1 9 8
first even number: 2, last even number: 8
new vector: 2 7 2 1 9
```

LẬP TRÌNH ĐA TIẾN TRÌNH

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
    // Store the value argument passed to this thread
    int *myid = (int *)vargp;

    // Let us create a static variable to observe its changes
    static int s = 0;

    // Change static and global variables
    ++s; ++g;

    // Print the argument, static and global variables
    printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
    int i;
    pthread_t tid;

    // Let us create three threads
    for (i = 0; i < 3; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)i);

    pthread_exit(NULL);
    return 0;
}
```

```
gfg@ubuntu:~/S gcc multithread.c -lpthread
gfg@ubuntu:~/S ./a.out
Thread ID: 1, Static: 1, Global: 1
Thread ID: 0, Static: 2, Global: 2
Thread ID: 2, Static: 3, Global: 3
gfg@ubuntu:~/S
```

LẬP TRÌNH ĐA TIẾN TRÌNH

```
#include <iostream>
#include <cstdlib>
#include <pthread.h>

using namespace std;

#define NUM_THREADS 5

void *PrintHello(void *threadid) {
    long tid;
    tid = (long)threadid;
    cout << "Hello World! Thread ID, " << tid << endl;
    pthread_exit(NULL);
}

int main () {
    pthread_t threads[NUM_THREADS];
    int rc;
    int i;

    for( i = 0; i < NUM_THREADS; i++ ) {
        cout << "main() : creating thread, " << i << endl;
        rc = pthread_create(&threads[i], NULL, PrintHello, (void *)i);

        if (rc) {
            cout << "Error:unable to create thread," << rc << endl;
            exit(-1);
        }
    }
    pthread_exit(NULL);
}
```

Compile the following program using -lpthread library as follows –

```
$gcc test.cpp -lpthread
```

Now, execute your program which gives the following output –

```
main() : creating thread, 0
main() : creating thread, 1
main() : creating thread, 2
main() : creating thread, 3
main() : creating thread, 4
Hello World! Thread ID, 0
Hello World! Thread ID, 1
Hello World! Thread ID, 2
Hello World! Thread ID, 3
Hello World! Thread ID, 4
```


LẬP TRÌNH PARALLEL THƯ VIỆN omp.h

```
gcc -fopenmp hellosmp.c -o hellosmp
```

```
//compute the sum of two arrays in parallel
#include <stdio.h>
#include <omp.h>
#define N 1000000
int main(void) {
    float a[N], b[N], c[N];
    int i;

    /* Initialize arrays a and b */
    for (i = 0; i < N; i++) {
        a[i] = i * 2.0;
        b[i] = i * 3.0;
    }

    /* Compute values of array c = a+b in parallel. */
    #pragma omp parallel shared(a, b, c) private(i)
    {
        #pragma omp for
        for (i = 0; i < N; i++) {
            c[i] = a[i] + b[i];
            printf ("%f\n", c[10]);
        }
    }
}
```

```
int main (int argc, char *argv[]) {
    int th_id, nthreads;

    #pragma omp parallel private(th_id)

    //th_id is declared above. It is specified as private: so each
    //thread will have its own copy of th_id
    {
        th_id = omp_get_thread_num();
        printf("Hello World from thread %d\n", th_id);
    }
}
```

```
#include <stdio.h>
#include <omp.h>
int main() {

    const int N=100;
    int a[N];

    //initialize
    for (int i=0; i<N; i++)
        a[i] = i;

    //compute sum
    int local_sum, sum;
    #pragma omp parallel private(local_sum) shared(sum)
    {
        local_sum =0;

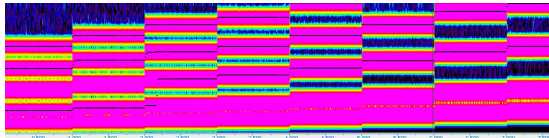
        //the array is distributed statically between threads
        #pragma omp for schedule(static,1)
        for (int i=0; i<N; i++) {
            local_sum += a[i];
        }

        //each thread calculated its local_sum. All threads have to add to
        //the global sum. It is critical that this operation is atomic.

        #pragma omp critical
        sum += local_sum;
    }

    printf("sum=%d should be %d\n", sum, N*(N-1)/2);
}
```

Program sound processing



```

1  #include "wave.h"
2  #include "writewave.c"
3  int ReadWave(char *filename, RiffHeader *R, FormatChunk *F, DataChunk *D);
4  int WriteWave(char *name,
5               short BitsPerSample, // 8bits 16bits 32bits
6               long SamplesPerSec,
7               short Channel,
8               unsigned char *Waveform_data,
9               long waveform_data_size
10             );
11
12
13  #define PI 3.14159265358979323846
14
15  unsigned char Harmonics(int type, double f, double t)
16  {
17      double value;
18
19      switch(type)
20      {
21          case 0:
22              value = 50.0*sin(2.0*PI*f*t);
23              break;
24          case 1:
25              value = 50.0*sin(2.0*PI*f*t)
26                    + 30.0*sin(2.0*PI*(2.0*f)*t)
27                    + 20.0*sin(2.0*PI*(3.0*f)*t)
28                    + 10.0*sin(2.0*PI*(4.0*f)*t)
29                    + 5.0*sin(2.0*PI*(5.0*f)*t);
30              break;
31          case 2:

```

MULTIMEDIA PROGRAMMING

```

48 void main()
49 {
50     // 264.0, 287.0, 330.0, 352.0, 396.0, 440.0, 495.0, 528.0
51     static double freq[]={
52         264.0, 287.0, 330.0, 352.0, 396.0, 440.0, 495.0, 528.0
53     };
54     char name[100];
55     long SamplesPerSec = 11025; // 11kHz sampling
56     short BitsPerSample = 8; // 8bits
57     short Channels = 1; // mono
58     double SamplesPeriod = 1/SamplesPerSec;
59     double PlayTime=1;
60     long waveformDataSize;
61     unsigned char *waveformData;
62     double t, f;
63     long index, length;
64     int i, num, type;
65
66     num = 8; // 8 channels
67
68     length = (long)(PlayTime*SamplesPerSec*Channels*(BitsPerSample/8));
69     waveformDataSize = length * num;
70     waveformData = (unsigned char *)malloc(sizeof(char)*waveformDataSize);
71
72     type = 3;
73     for(i=0;i<num;i++){
74         f = freq[i]; t=0.0;index=length;index++, t+=1.0/SamplesPerSec
75         waveformData[length*i+index] = Harmonics(type, f, t);
76     }
77 }

```

Đồ Rè Mi Fa Sol La Si Đồ

```

1  #include "wave.h"
2
3  void LongToString(long H_chunkID, char *chunkID)
4  {
5      long MASK = 0x000000FF;
6
7      chunkID[0] = (char)(H_chunkID & MASK);
8      chunkID[1] = (char)((H_chunkID >> 8) & MASK);
9      chunkID[2] = (char)((H_chunkID >> 16) & MASK);
10     chunkID[3] = (char)((H_chunkID >> 24) & MASK);
11     chunkID[4] = '\0';
12 }
13
14
15 int ReadWave(char *filename, RiffHeader *R, FormatChunk *F, DataChunk *D) // read wave
16 {
17     FILE *fp;
18     HEADER H;
19     char chunkID[5];
20
21     if( (fp=fopen(filename,"rb")) == NULL ) {
22         printf("\n\t Reading File Not Found!!\n");
23         return(0);
24     }
25
26     // WAVE Header File Information
27
28     while(0 != fread(&H, sizeof(HEADER), 1, fp)){
29         LongToString(H.chunkID, chunkID);
30
31         if(strcmp(chunkID, "RIFF")==0){
32             R->chunkID = H.chunkID;
33             R->chunkSize = H.chunkSize;
34             fread(&R->field.vFormat, sizeof(R->field.vFormat), 1, fp);
35             LongToString(R->field.vFormat, chunkID);
36
37             if(strcmp(chunkID, "WAVE")==0){
38                 printf("\n\t\t Not supported format !!");
39                 return(0);
40             }
41         }
42         else if(strcmp(chunkID, "fmt ")==0){
43             R->chunkID = H.chunkID;
44             R->chunkSize = H.chunkSize;
45             fread(&R->field, sizeof(R->field), 1, fp);
46             if(R->field.wBitsPerSample == 8) || (R->field.wBitsPerSample == 16){
47                 printf("\n\t\t BitsPerSample = %d is not supported !!", R->field.wBitsPerSample);
48                 return(-1);
49             }
50             if(H.chunkSize==sizeof(R->field)){
51                 fseek(fp, (long)(H.chunkSize-sizeof(R->field)), SEEK_CUR);
52             }
53             else if(strcmp(chunkID, "data")==0){
54                 D->chunkID = H.chunkID;
55                 D->chunkSize = H.chunkSize;
56                 D->field.waveformData = (unsigned char *)malloc(sizeof(char)*H.chunkSize);
57                 if(D->field.waveformData==NULL) return(-1);
58                 fread(D->field.waveformData, 1, H.chunkSize, fp);
59             }
60         }
61     }
62 }

```

READ WAVE FILE

```

63     else( // RIFF, data, data 88 chunkID 32WU
64         fseek(fp, (long)H.chunkSize, SEEK_CUR);
65     )
66 }
67
68 fclose(fp);
69
70 return(1);
71 }
72

```

```

3 int WriteWave2(char *filename, RiffHeader R, FormatChunk F, DataChunk D)
4 {
5     FILE *fp;
6
7     if((fp=fopen(filename, "wb")) == NULL){
8         printf("\t File Open Failure!!\n");
9         return(0);
10    }
11
12    fwrite(&R, sizeof(R), 1, fp);
13
14    F.chunkSize = 16;
15    fwrite(&F, sizeof(F), 1, fp);
16
17    fwrite(&D.chunkID, sizeof(D.chunkID), 1, fp);
18    fwrite(&D.chunkSize, sizeof(D.chunkSize), 1, fp);
19    fwrite(D.field.waveformData, sizeof(char), D.chunkSize, fp);
20
21    fclose(fp);
22
23    return(1);
24 }
25
26 int WriteWave(char *name,
27               short BitsPerSample, // Shift 16 bits to the left
28               long SamplesPerSec,
29               short Channel,
30               unsigned char *waveform_data,
31               long waveform_data_size
32 )

```

Write wave File

```

33 {
34     RiffHeader R;
35     FormatChunk F;
36     DataChunk D;
37
38     R.chunkID = 0x46644952; // "RIFF"
39     R.chunkSize = 16 + waveform_data_size + 20;
40     R.field.wFormat = 0x5564157; // "WAVE"
41
42     F.chunkID = 0x20746d66; // "fmt "
43     F.chunkSize = 16;
44     F.field.wFormatTag = 1;
45     F.field.wChannels = Channel;
46     F.field.dwAvgBytesPerSec = SamplesPerSec * Channel * (BitsPerSample/8);
47     F.field.dwSamplesPerSec = SamplesPerSec;
48     F.field.wBitsPerSample = BitsPerSample;
49     F.field.wBlockAlign = Channel * (BitsPerSample/8);
50
51     D.chunkID = 0x61746164; // "data"
52     D.chunkSize = waveform_data_size;
53
54     D.field.waveformData = waveform_data;
55
56     WriteWave2(name, R, F, D);
57
58     return(1);
59 }
60

```

FORMAT WAVE FILE

```

5 #include <math.h>
6
7 #define ID long
8
9 typedef struct {
10     ID chunkID;
11     long chunkSize;
12 } HEADER;
13
14 typedef struct {
15     long wFormat;
16 } subRiffHeader;
17
18 typedef struct {
19     ID chunkID;
20     long chunkSize;
21     subRiffHeader field;
22 } RiffHeader;
23
24 typedef struct {
25     short wFormatTag; // PCM = 1, Values other than 1 indicate some form of compression
26     unsigned short wChannels; // the number of audio channels (Mono = 1, Stereo = 2, etc)
27     unsigned long dwSamplesPerSec; // sampling rate, or sample frame per sec
28     unsigned long dwAvgBytesPerSec; //the number of bytes (xxxxxx) @*, 2*(dwSamplesPerSec)
29     unsigned short wBlockAlign; //the size of a sample frame, in terms of bytes: wBlockAlign = wChannels
30     unsigned short wBitsPerSample; //the bit resolution of a sample point
31     /* Note: there may be additional fields here, depending upon wFormatTag. */
32 } subFormatChunk;
33
34 typedef struct {
35     ID chunkID; //ASCIIIZE "fmt" (0x666d7420 big-endian form).
36     long chunkSize; //16 for PCM the size of the rest of the chunk which follows this number
37     subFormatChunk field;
38 } FormatChunk;
39
40 typedef struct {
41     unsigned char *waveformData;
42 } subDataChunk;
43
44 typedef struct {
45     ID chunkID; //ASCIIIZE "data" (0x64617461 big-endian form).
46     long chunkSize; //the number of remaining bytes in the chunk after the chunkSize field
47     subDataChunk field;
48 } DataChunk;
49

```

VIEW JPEG IMAGE

```

59 class Cjpeg
60 {
61 public:
62     Cjpeg();
63     virtual ~Cjpeg();
64 private:
65     //*****
66     // Number of
67     //*****
68     WORD Ri; // Restart Interval
69     int m_rWidth; // Restart Interval
70     int m_rHeight; // Restart Interval
71     BYTE * pByte; // NextByte()
72     int cnt; // Count of
73     short *Y; // Save Y Buffer
74     short *Cb; // Save Cb Buffer
75     short *Cr; // Save Cr Buffer
76     SET *MCU; // MCU
77     BYTE Rmax; // Maximum Horizontal Sampling Factor
78     BYTE Vmax; // Maximum Vertical Sampling Factor
79     BYTE *m_pData; // Data
80     BYTE *m_pBuf; // Buffer
81     int m_Index; // Index
82     DOT Tbl[20]; // Quantization Table
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97 // RGB Color YCbCr Color Space. //
98 float R, G, B;
99 float y, cb, cr;
100 BYTE *pos;
101
102 for(i=0; i<Height; i++)
103 {
104     pos = m_pData[i*(bWidth*3)];
105     for(j=0; j<Width; j++)
106     {
107         B = (float)*pos;
108         G = (float)*(pos+1);
109         R = (float)*(pos+2);
110
111         y = 0.299f * R + 0.587f * G + 0.114f * B;
112         cb = -0.1687f * R - 0.3313f * G + 0.5f * B + 128;
113         cr = 0.5f * R - 0.4187f * G - 0.0813f * B + 128;
114
115         /* y = (77 * R + 150 * G + 29 * B) >> 8;
116         cb = (B - y) + 128;
117         cr = (R - y) + 128; */
118
119         if(y>255.) y = 255.; if(y<0.) y = 0.;
120         if(cb>255.) cb = 255.; if(cb<0.) cb = 0.;
121         if(cr>255.) cr = 255.; if(cr<0.) cr = 0.;
122
123         *pos = (BYTE)y;
124         *(pos+1) = (BYTE)cb;
125         *(pos+2) = (BYTE)cr;
126     }
127 }

```

THƯ VIỆN DOS

- Các hàm kiểm soát thư mục
 - int chdir(char *s) : đổi thư mục chủ
- Lấy tên thư mục chủ
 - char *getcwd(char *s, int n)
- Tạo thư mục mới
 - int mkdir (char *s)
- Xóa thư mục
 - rmdir(char *s)
- Tìm tệp trên thư mục
 - int findfirst (char *path, struct fblk *fb, int attrib)
- Tiếp tục tìm kiếm trên thư mục
 - findnext(struct fblk *fb)

Hiện lên màn hình danh sách các tệp trong 1 thư mục: dùng hàm
findfirst và findnext

```
#include "dir.h"
#include "dos.h"
#include "stdio.h"
#include "conio.h"

char bs[] = "\\*.*";
#define attr (
FA_RDONLY|FA_HIDDEN|FA_SYSTEM|FA_LABEL|FA_DIRECT|FA_ARCH)

int sf, nn;
void scandir (char *d)
{
    char dd[30], ddd[30];
    int first = 1;
    struct ffblk f; int s;
    sprintf(dd, "%s%s", d, bs);
    printf("\n\n%s", ss);
}
```

```
while(1)
{
    if (first)
    {
        s=findfirst(dd, &f, attr);
        first =0;
    }
    else
    s= findnext(&f);
    if (s!=0)
    return 0;
    if (f.ff_name[0]!='.') continue;
    if (f.attrib==FA_DIRECT)
    {
        sprintf(ddd, "%s\\%s",d,f.ff_name);
        scandir(ddd);
    }
    else
    {
        ++sf;
        printf("\n Ten %s", f.ff_name);
        if (sf %20 ==0)
        { printf("\n bam enter de xem"); getch();
        }
    }
}
```

```
}
}
}
int main ()
{
    char d[30];
    clrscr();
    printf("n cho biet thu muc can xet");
    gets(s);
    scandir(d);
    getch();
    clrscr();
    return 1;
}
```

CÁC HÀM KIỂM SOÁT QUÁ TRÌNH

```
#include <stdio.h>
#include <process.h>
#include <ctype.h>

int main ()
{
    system ("cls");
    sytem ("dir *.bak");
    puts("Có xóa ? C/K");
    if (toupper (getch())=='C')
    {
        system ("del *.bak");
        system("dir *.bak");
    }
    else
    exit (0);
    getch();
}
```

- Là 2 tham số : argc và argv
- Tham số argc là số nguyên chỉ tham số trên dòng lệnh, có giá trị nhỏ nhất =1, vì bản thân tên chương trình là tham số thứ nhất
- Tham số argv là mảng các con trỏ, trỏ đến các tham số trên dòng lệnh: char *argv[];

SỬ DỤNG HÀM MAIN CÓ ĐỐI

argv[0]: chứa địa chỉ của tên chương trình

argv[1]: chứa địa chỉ của tham số thứ nhất

argv[2]: chứa địa chỉ của tham số thứ hai

Ví dụ: Chương trình sau đã được biên dịch thành MYPRO.EXE, nếu nhập trên dòng lệnh MYPRO thì có dòng nhắc nhở, nếu nhập MYPRO LAN thì Chao ban LAN

```
#include <stdio.h>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    if (argc !=2) printf("Phai nhap Ten");
```

```
    else printf("Chao ban %s\n",argv[1]);
```

```
    return 1;
```

```
}
```

CHIA SẺ CHỨC NĂNG CHƯƠNG TRÌNH GIỮA CÁC NGÔN NGỮ

Cách viết 1

Java program

Build code

```
javac hello_world.java
Java hello
```

System ("ngon ngu C")

Cách viết 2

C program

Build code

System ("ngon ngu Java")