



BÀI THỰC HÀNH SỐ 04

1. MỤC TIÊU

Hiểu và cài đặt được chương trình có sử dụng kiểu dữ liệu cấu trúc (struct).

2. LÝ THUYẾT CẦN GHI NHỚ

2.1. Khái niệm

Kiểu cấu trúc (*Structure*) là một kiểu dữ liệu do người dùng tự định nghĩa, là kiểu dữ liệu bao gồm nhiều thành phần, mỗi thành phần có một kiểu dữ liệu khác nhau, mỗi thành phần được gọi là một trường (field).

2.2. Khai báo kiểu cấu trúc

2.2.1. Khai báo

```
typedef struct <Tên cấu trúc>
{
    <kiểu dữ liệu> <trường 1> ;
    < kiểu dữ liệu > < trường 2> ;
    ...
    < kiểu dữ liệu > < trường n> ;
};
```

Trong đó:

- <Tên cấu trúc>: là tên của kiểu cấu trúc sẽ được dùng. Tên này phải được đặt theo quy tắc đặt tên cho các định danh trong C.
- < kiểu dữ liệu > < trường i> (i=1..n): mỗi trường trong cấu trúc có dữ liệu thuộc kiểu gì. Tên của trường phải là một tên được đặt theo quy tắc đặt tên cho các định danh trong C.
- Ví dụ 1: Khai báo một kiểu struct có tên **NgayThang** có các thành phần là: ngày, tháng, năm.

```
typedef struct DATE
{
    unsigned char  ngay;
    unsigned char  thang;
    int            nam;
};
```

- Ví dụ 2: Khai báo một kiểu struct có tên **KETQUA** có các thành phần là: Mã số sinh viên, Mã số môn học, Lần thi, điểm thi.

```
struct KETQUA
{
    char masv[8]; /* mã số sinh viên */
    char mamh[5]; /* mã số môn học */
    int  lanthi;  /* lần thi */
    float diem;   /* điểm thi */
};
```



- Ví dụ 3: Định nghĩa cấu trúc dữ liệu cho phân số

```
typedef struct phanso
{
    int tu, mau ;
};
```

Khi đó Nếu ta khai báo một biến a có kiểu dữ liệu là phân số thì a gồm 2 thành phần: a.tu, a.mau

PS a

a.tu	a.mau
------	-------

2.2.2. Khai báo và khởi tạo giá trị cho biến cấu trúc

- Cách khởi tạo giá trị cho biến cấu trúc giống như khởi tạo cho mảng.

- Ví dụ 4: DATE birth ={12,10,2000};
 KETQUA kq={"1234567", "Toan",1,4.5};

2.2.3. Kiểu dữ liệu có cấu trúc có thể lồng vào nhau.

Ví dụ 5: Định nghĩa kiểu dữ liệu của học sinh HOCSINH gồm:

- ▣ Mã số học sinh (MSHS) : chuỗi có tối đa 5 ký tự.
- ▣ Họ tên (hoten) : chuỗi có tối đa 30 ký tự.
- ▣ Ngày tháng năm sinh (ngaysinh) : kiểu DATE.
- ▣ Địa chỉ (diachi) : chuỗi có tối đa 50 ký tự.
- ▣ Giới tính (phai) : chuỗi có tối đa 3 ký tự.
- ▣ Điểm trung bình (diemtb) : số thực.

struct HOCSINH được định nghĩa như saunhư sau:

```
struct DATE
{
    unsigned char ngay, thang;
    int nam;
};
typedef struct HOCSINH
{
    char MSHS[6];
    char hoten[31];
    struct DATE ngaysinh;
    char diachi[51];
    unsigned char phai[4];
    float diemtb;
};
```

⚠ Khi định nghĩa kiểu dữ liệu struct lồng nhau, cần lưu ý: Kiểu dữ liệu cấu trúc được sử dụng phải khai báo trước (đặt ở phía trên).



2.3. Sử dụng biến cấu trúc

2.3.1. Khai báo biến kiểu cấu trúc

Khi định nghĩa kiểu dữ liệu tức là ta có một kiểu dữ liệu mới, muốn sử dụng ta phải khai báo biến. Cú pháp khai báo biến của kiểu dữ liệu mới định nghĩa cũng giống như cách khai báo của các kiểu dữ liệu chuẩn.

`struct <tên cấu trúc> <tên biến> ;`

Ví dụ 6: `struct DATE x ; // Khai báo biến x có kiểu dữ liệu DATE`

Tuy nhiên nếu khi định nghĩa struct có dùng từ khoá typedef thì ta có thể khai báo trực tiếp mà không cần từ khoá struct.

Ví dụ 7: `DATE x ; // Khai báo biến x có kiểu DATE`

2.3.2. Truy cập vào từng phần tử của cấu trúc

- Cú pháp

Thông qua dấu chấm để truy cập đến từng thành phần của struct .

`<tên biến struct>.<tên thành phần của struct>`

- *Ví dụ 8:* Để xuất ra màn hình tên của học sinh viên hs1, ta thực hiện câu lệnh sau:

```
HOCSINH hs1;
printf("Nhap diem trung binh:");
printf("%f", &hs1.diemtb);
...
printf("Diem trung binh= %.2f", hs1.diemtb);
```

2.4. Mảng cấu trúc

2.4.1. Khai báo

Tương tự như khi khai báo mảng với kiểu dữ liệu chuẩn của C. Chỉ khác là thay kiểu dữ liệu chuẩn bằng kiểu dữ liệu có cấu trúc.

Ví dụ 9 `SINHVIEN A[SIZE];`

`DIEM P[200];`

2.4.2. Truy xuất phần tử trong mảng

Tương tự như truy cập trên mảng một chiều hay ma trận. Nhưng do từng phần tử có kiểu cấu trúc nên phải chỉ định rõ cần lấy thành phần nào, tức là phải truy cập đến thành phần cuối cùng có kiểu là dữ liệu cơ bản.

Ví dụ 10 `SINHVIEN A[SIZE];`

`...`

`A[i].DiemTB=7.25;`

`A[j].DiemTB=A[i].DiemTB;`




Hay DIEM P[200];

...

P[j].x=1357;

P[j].y=P[i].x * 2;

2.5. Sử dụng cấu trúc để trừu tượng hóa dữ liệu

 Phương pháp luận giải quyết bài toán bằng phương pháp trừu tượng hóa dữ liệu:
(sử dụng phương pháp lập trình hướng hàm hàm)

Bước 1. Xác định các kiểu dữ liệu. Trong bước này ta phải xác định xem trong bài toán (vấn đề) phải làm việc với những kiểu dữ liệu nào. Kiểu dữ liệu nào đã có sẵn, kiểu dữ liệu nào phải định nghĩa mới.

Bước 2. Thiết kế hàm. Trong bước này ta phải xác định xem trong bài toán mà ta giải quyết cần phải có bao nhiêu hàm, tên của các hàm ra sao, các tham số đầu vào, kiểu dữ liệu trả như thế nào. Quan trọng hơn nữa là các tham số thì tham số nào là tham trị và tham số nào là tham biến.

Bước 3. Định nghĩa hàm. Trong bước này ta sẽ tiến hành định nghĩa các hàm đã thiết kế và khai báo trong bước 2. Hơn nữa ta phải định nghĩa hàm main với các khai báo biến và thực hiện các lời gọi hàm cần thiết cho chương trình.

Lưu ý: Bước 1 và Bước 2 là hai bước quan trọng nhất.

3. BÀI TẬP CÓ HƯỚNG DẪN

3.1. Thao tác trên biến đơn kiểu cấu trúc

3.1.1. Yêu cầu

Khai báo kiểu dữ liệu biểu diễn khái niệm phân số và định nghĩa các hàm:

- (i). Nhập 1 phân số: cần thực hiện:
 - Rút gọn phân số, ví dụ 4/8 được rút gọn thành 1/2.
 - Khi mẫu số là số âm, cần khử dấu trừ của mẫu số bằng cách nhân cả tử và mẫu cho -1.
 - Khi tử số = 0 thì mẫu số sẽ được gán = 1.
- (ii). Xuất 1 phân số dưới dạng tử/mẫu, ví dụ -2/3.
- (iii). Tính tổng, hiệu, tích, thương của 2 phân số. Sau khi tính toán cũng cần rút gọn phân số kết quả.
- (iv). So sánh 2 phân số,



3.1.3. Thực hiện

- Khai báo cấu trúc PHANSO

```
struct PHANSO
{
    int tu;
    int mau;
};
```

- Viết hàm nhập 1 cấu trúc PHANSO và các hàm liên quan đến việc rút gọn phân số

```
void NhapPS (PHANSO &p)
{
    printf("\nNhap tu so: ");
    scanf("%d", &p.tu);
    do
    {
        printf("Nhap mau so (!=0): ");
        scanf("%d", &p.mau);
        if (p.mau == 0)
            printf("Mau so phai khac zero (0)\n");
    } while (p.mau == 0);
    RutGonPS (p);
}

int TimUSCLN(int a, int b)
{
    a=abs(a);
    b=abs(b);
    while (a != b)
        if (a>b)
            a -= b;
        else
            b -= a;
    return a;
}

void RutGonPS (PHANSO &p)
{
    if (p.tu != 0)
    {
        int usc = TimUSCLN(abs(p.tu), abs(p.mau));
        p.tu /= usc;
        p.mau /= usc;
        // nếu mẫu là số âm thì khử dấu trừ của mẫu số
        if (p.mau < 0)
        {
            p.tu *= -1;
            p.mau *= -1;
        }
    }
    else
        p.mau = 1;
}
```



- Viết hàm xuất 1 phân số

```
void XuatPS (PHANSO x)
{
    printf("%d/%d", x.tu, x.mau);
}
```

- Viết hàm tính tổng 2 phân số

```
PHANSO Cong2PhanSo (PHANSO x, PHANSO y)
{
    PHANSO C;
    C.mau = x.mau*y.mau;
    C.tu = (x.tu*y.mau) + (y.tu*x.mau);
    RutGonPS (C);
    return C;
}

void Tong (PHANSO x, PHANSO y)
{
    PHANSO C;
    C = Cong2PhanSo (A, B);
    XuatPS (A);
    printf(" + ");
    XuatPS (B);
    printf(" = ");
    XuatPS (C);
}
```

- Khai báo các nguyên mẫu hàm và nội dung hàm main

```
void NhapPS (PHANSO &x);
void XuatPS (PHANSO x);
int TimUSCLN (int a, int b);
PHANSO Cong2PhanSo (PHANSO x, PHANSO y);
void RutGonPS (PHANSO &p);
void Tong (PHANSO x, PHANSO y);
int main()
{
    PHANSO A, B, C;
    printf("Nhap phan so thu 1");
    NhapPS (A);
    printf("Nhap phan so thu 2");
    NhapPS (B);
    Tong (A, B);
    return 0;
}
```

- Dựa trên hàm Tổng đã có, sinh viên tự triển khai các hàm còn lại: hiệu, tích, thương, so sánh để hoàn tất yêu cầu



3.2. Thao tác trên mảng cấu trúc

3.2.1. Yêu cầu

Tổ chức dữ liệu quản lý xe gắn máy với số lượng (n) dự kiến nằm trong khoảng $0 \leq n \leq 100$. Các thông tin liên quan đến mỗi xe gồm:

- Số xe (kiểu chuỗi, tối đa 12 ký tự).
- Số Km đã đi được (kiểu số thực).
- Hãng sản xuất (kiểu số nguyên). Quy ước: 0-Honda, 1-Yamaha, 2-Vespa, 3-Suzuki.
- Động cơ xăng (kiểu bool). Quy ước: true-động cơ dùng xăng; false- động cơ dùng xăng pha nhớt.

Viết chương trình gồm các hàm chức năng để thực hiện những công việc sau:

- a. Hàm nhập vào 1 xe.
- b. Hàm nhập danh sách n xe. Hàm này sẽ gọi hàm nhập 1 xe ở câu a thực hiện.
- c. Hàm xuất thông tin về 1 xe.
- d. Hàm xuất danh sách n xe. Hàm này sẽ gọi hàm xuất 1 xe ở câu c thực hiện
- e. Hàm xuất thông tin các xe của hãng X, có loại động cơ là Y. Với X và Y là 2 tham số đầu vào.

3.2.2. Thực hiện

- Khai báo đầu chương trình

```
#include <iostream>
#include <conio.h>
#define SIZE 100
```

```
typedef struct XE
```

```
{
```

```
    char SoXe[12];
```

```
    float SoKm;
```

```
    int HangSanXuat;    //0-Honda, 1-Yamaha, 2-Vespa, 3-Suzuki
```

```
    bool DongCoXang;    /*true-động cơ dùng xăng;
```

```
                        false- động cơ dùng xăng pha nhớt */
```

```
};
```

- Viết hàm thực hiện yêu cầu của câu a (Hàm nhập vào 1 xe)

```
void Nhap1Xe(XE &x)
```

```
{
```

```
    printf("\nNhap so xe: ");
```

```
    fflush(stdin);
```

```
    gets(x.SoXe);
```

```
    printf("Nhap so Km: ");
```

```
    scanf("%f", &x.SoKm);
```

```
    printf("Nhap hang san xuat (0-Honda, 1-Yamaha, 2-Vespa, 3-Suzuki): ");
```

```
    scanf("%d", &x.HangSanXuat);
```

```
    printf("Dong co xang (Y/N): ");
```



```
fflush(stdin);  
char VietNam=getchar();  
if ((VietNam=='Y') || (VietNam=='y'))  
    x.DongCoXang=true;  
else  
    x.DongCoXang=false;  
}
```

- Viết hàm thực hiện yêu cầu của câu b (Hàm nhập danh sách n xe)

```
int NhapN()  
{  
    int n;  
    do  
    {  
        printf("Nhap n (0<n<=%d): ", SIZE);  
        scanf("%d", &n);  
        if ((n<=0) || (n>SIZE))  
            printf("Chi nhan cac so tu 1 den %d.  
                    Nhap lai.\n", SIZE);  
    }while ((n<=0) || (n>SIZE));  
    return n;  
}  
void NhapMang(XE A[], int n)  
{    for (int i=0; i<n; i++)  
        Nhap1Xe(A[i]);  
}
```

- Viết hàm thực hiện yêu cầu của câu c (Hàm xuất thông tin về 1 xe)

```
void Xuat1Xe(XE x)  
{  
    printf("\nSo xe: %s", x.SoXe);  
    printf("\nSo Km da di: %f", x.SoKm);  
    if (x.HangSanXuat==0)  
        printf("\nHang san xuat: Honda");  
    else  
        if (x.HangSanXuat==1)  
            printf("\nHang san xuat: Yamaha");  
        else  
            if (x.HangSanXuat==2)  
                printf("\nHang san xuat: Vespa");  
            else  
                printf("\nHang san xuat: Suzuki");  
    if (x.DongCoXang==true)  
        printf("\nXe dong co xang");  
    else  
        printf("\nXe dong co xang pha nhot");  
}
```




- Viết hàm thực hiện yêu cầu của câu d (Hàm xuất danh sách n xe)

```
void XuatMang(XE A[],int n)
{
    for (int i=0;i<n;i++)
        Xuat1Xe(A[i]);
}
```

- Viết hàm thực hiện yêu cầu của câu e (Hàm xuất thông tin các xe của hãng X, có loại động cơ là Y. Với X và Y là 2 tham số đầu vào)

```
void XuatXeTheoDieuKien(XE A[],int n, int hangXe, bool dongCo)
{
    printf("Cac xe cua hang ");
    if (hangXe==0)
        printf("Honda");
    else
        if (hangXe==1)
            printf("Yamaha");
        else
            if (hangXe==2)
                printf("Vespa");
            else
                printf("Suzuki");
    printf(" su dung dong co ");
    if (dongCo==true)
        printf("xang:\n");
    else
        printf("xang pha nhot:\n");
    for (int i=0;i<n;i++)
        if ((A[i].HangSanXuat==hangXe)
            && (A[i].DongCoXang==dongCo))
            Xuat1Xe(A[i]);
}
```

- Hàm main

```
int main()
{
    XE A[SIZE];
    int n, hangXe;
    bool dongCo;
    n=NhapN();
    NhapMang(A,n);
    printf("\nDanh sach xe vua nhap: \n") ;
    XuatMang(A,n);
    printf("\nNhap hang san xuat can xem (0-Honda, 1-Yamaha,
        2-Vespa, 3-Suzuki): ");
    scanf("%d",&hangXe);
    printf("Xe su dung dong co xang (Y/N): ");
    fflush(stdin);
    char VietNam=getchar();
}
```



```
if ((VietNam=='Y') || (VietNam=='y'))
    dongCo=true;
else
    dongCo=false;
XuatXeTheoDieuKien(A, n, hangXe, dongCo);
return 0;
}
```

- Khai báo nguyên mẫu hàm

```
int NhapN();
void Nhap1Xe(XE &x);
void NhapMang(XE A[],int n);
void Xuat1Xe(XE x);
void XuatMang(XE A[],int n);
void XuatXeTheoDieuKien(XE A[],int n, int hangXe, bool dongCo);
```

4. BÀI TẬP DO SINH VIÊN TỰ THỰC HIỆN

Kiểu cấu trúc đơn

Bài 1. Từ bài phân số trong phần 3.1.3, yêu cầu bổ sung như sau:

- Tạo một mảng các phân số: void NhapMangPS(PHANSO A[], int n). Hàm này sẽ gọi hàm nhập 1 phân số (Nhap1PS) đã có.
- Xuất mảng phân số: void XuatMangPS(PHANSO A[], int n). Hàm này sẽ gọi hàm xuất 1 phân số đã có.
- Tìm phân số lớn nhất, phân số nhỏ nhất. PHANSO TimPSMax(PHANSO A[], int n)
- Sắp xếp mảng phân số tăng dần: void SortMangPS(PHANSO A[], int n)
- Cho biết tổng, hiệu, tích, thương của 2 phân số lớn nhất và phân số nhỏ nhất đang có trong mảng.

Bài 2. Tổ chức dữ liệu quản lý danh mục các bộ phim. Các thông tin liên quan đến mỗi bộ phim gồm:

- Tên phim (kiểu chuỗi, tối đa 50 ký tự).
- Doanh thu (kiểu số thực).
- Thể loại (kiểu số nguyên). Quy ước: 0-hình sự, 1-tình cảm, 2-hài.
- Năm sản xuất (kiểu số nguyên).
- Phim Việt Nam sản xuất (kiểu bool). Quy ước: true-Việt Nam; false-nước ngoài.

Viết chương trình gồm các hàm chức năng để thực hiện những công việc sau:

- Hàm nhập vào 1 bộ phim mới cùng với các thông tin liên quan đến bộ phim này.
- Hàm nhập danh sách n phim, với n do người dùng nhập trực tiếp trong hàm.
- Hàm xuất thông tin về 1 bộ phim.



- d. Hàm xuất danh sách n phim đang có trong danh sách.
- e. Hàm cho nhập một thể loại: In ra danh sách các bộ phim thuộc thể loại này.
- f. Hàm tính tổng doanh thu của các phim do Việt Nam sản xuất.
- g. Hàm nhận 2 tham số nguyên là *fromYear* (từ năm) và *toYear* (đến năm). Hàm thực hiện tính doanh thu trung bình của tất cả các phim năm sản xuất nằm trong khoảng từ *fromYear* đến *toYear*.
- h. Hàm nhận 2 tham số nguyên là *LowerBound* (cận dưới của doanh thu) và *UpperBound* (cận trên của doanh thu). Hàm thực hiện đếm số lượng phim có doanh thu thấp hơn *LowerBound* hoặc lớn hơn *UpperBound*.

Bài 3. Tổ chức dữ liệu quản lý danh mục các máy tính của một cửa hàng. Các thông tin liên quan đến mỗi máy tính gồm:

- Model (kiểu chuỗi, tối đa 10 ký tự).
- Hãng sản xuất (kiểu chuỗi, tối đa 30 ký tự).
- Thời gian bảo hành (kiểu số nguyên). Quy ước: **1: 3 tháng; 2: 6 tháng; 3: 12 tháng; 4: 24 tháng; 5: 36 tháng.**
- Giá tiền (kiểu float)
- Sản xuất tại Việt Nam (kiểu bool). Quy ước: true-Việt Nam; false-nước ngoài.

Viết chương trình gồm các hàm chức năng để thực hiện những công việc sau:

- a. Hàm nhập vào 1 máy tính mới cùng với các thông tin liên quan đến máy tính này. Cần kiểm tra giá trị nhập, nếu nhập sai, chương trình phải yêu cầu nhập lại cho đến khi nhập đúng. Các thông tin cần kiểm tra gồm:
 - Thời gian bảo hành phải nằm trong khoảng từ 1 đến 5.
 - Sản xuất tại Việt Nam: cho người dùng nhập Y hoặc y là sản xuất tại Việt Nam; nhập N hoặc n là sản xuất tại nước ngoài. Các ký tự được nhập phải là 1 trong các ký tự Y, y, N, n.
- b. Hàm nhập danh sách n máy tính, với n do người dùng nhập trực tiếp trong hàm.
- c. Hàm xuất thông tin về 1 máy tính. Yêu cầu:
 - Thời gian bảo hành phải thể hiện rõ số tháng bảo hành.
 - Sản xuất tại Việt Nam: phải ghi rõ “*Sản xuất tại Việt Nam*” hoặc “*Sản xuất tại nước ngoài*”.
- d. Hàm xuất các máy tính đang có trong danh sách.
- e. Hàm cho nhập hãng sản xuất: In ra danh sách các máy tính do hãng này sản xuất.
- f. Hàm nhận 2 tham số nguyên là *fromPrice* (từ giá) và *toPrice* (đến giá). Hàm thực hiện đếm số lượng model máy tính có giá tiền nằm trong khoảng từ *fromYear* đến *toYear*.
- g. Hàm nhận 2 tham số nguyên là *lowerPrice* (giá thấp nhất) và *upperPrice* (giá cao nhất). Hàm thực hiện in ra danh sách các máy tính do Việt Nam sản xuất và có giá tiền thấp hơn *lowerPrice* hoặc cao hơn *upperPrice*.



Bài 4. Để lắp ráp một máy vi tính hoàn chỉnh cần phải có tối thiểu 10 linh kiện loại A và có thể lắp bổ sung thêm vào khoảng tối đa 8 linh kiện loại B. Tại một cửa hàng vi tính cần quản lý bán hàng các loại linh kiện tại cửa hàng. Thông tin về một loại linh kiện gồm có: Tên linh kiện, quy cách, loại, đơn giá loại 1 (chất lượng tốt – số nguyên), đơn giá loại 2 (chất lượng thường – số nguyên). Viết chương trình thực hiện những công việc sau:

- Nhập vào thông tin về các linh kiện có ở cửa hàng.
- Xuất danh sách các linh kiện đã nhập theo thứ tự tăng dần của loại linh kiện và tên linh kiện.
- Cho biết đã có đủ 10 linh kiện loại A cần thiết lắp ráp máy hay chưa?

Bài 5. Một thư viện cần quản lý thông tin về các đầu sách. Mỗi đầu sách bao gồm các thông tin sau: MaSach (mã số sách), TenSach (tên sách), TacGia (tác giả), SL (số lượng các cuốn sách của đầu sách). Viết chương trình thực hiện các chức năng sau:

- Nhập vào một danh sách các đầu sách (tối đa là 100 đầu sách)
- Nhập vào tên của quyển sách. In ra thông tin đầy đủ về các sách có tên đó, nếu không có thì tên của quyển sách đó thì báo là: “*Không tìm thấy*”.
- Tính tổng số sách có trong thư viện.

Bài 6. Một cửa hàng cần quản lý các mặt hàng, thông tin một mặt hàng bao gồm:

- Mã hàng - Số lượng - Số lượng tồn
- Tên mặt hàng - Đơn giá - Thời gian bảo hành (tính theo tháng)
- Hãy nhập vào một danh sách các mặt hàng.
- Tìm mặt hàng có số lượng tồn nhiều nhất.
- Tìm mặt hàng có số lượng tồn ít nhất.
- Tìm mặt hàng có giá tiền cao nhất.
- In ra những mặt hàng có thời gian bảo hành lớn hơn 12 tháng.
- Sắp xếp các mặt hàng theo thứ tự tăng dần của số lượng tồn.

Bài 7. Viết chương trình quản lý hồ sơ nhân viên trong một công ty, chương trình thực hiện những công việc sau:

- Họ và tên. - Ngày sinh. - Lương cơ bản. - Thưởng.
- Phái. - Địa chỉ. - Bảo hiểm xã hội. - Phạt.
- $\text{Lương thực lĩnh} = \text{lương cơ bản} + \text{thưởng} - \text{BH xã hội} - \text{phạt}$.
- Nhập vào hồ sơ của các nhân viên trong công ty.
- Xuất danh sách các nhân viên theo lương thực lĩnh giảm dần bằng 2 cách sau:
 - Cấp phát vùng nhớ tĩnh.
 - Cấp phát vùng nhớ động.



Bài 8. Sử dụng cấu trúc để viết chương trình tính khoảng cách giữa 2 ngày.

Ví dụ ngày1= 15/11/2007

ngày2= 11/9/2009

in ra ngày 15/11/2007 và ngày 11/8/2009 cách nhau 26 ngày 9 tháng và 1 năm

Bài 9. Viết chương trình sử dụng con trỏ cấu trúc để hiển thị giờ, phút, giây ra màn hình, và tính khoảng cách giữa 2 mốc thời gian.