

Tổng quan Cấu trúc dữ liệu và giải thuật

GVGD: Trương Phước Hải

LOGO

Nội dung

- ◆ Vai trò của CTDL
- ◆ Tiêu chuẩn đánh giá CTDL
- ◆ Kiểu dữ liệu
- ◆ Con trỏ
- ◆ Mảng
- ◆ Chuỗi
- ◆ Cấu trúc

2

Vai trò của CTDL

◆ Vấn đề cần chú trọng khi chuyển bài toán thực tế thành mô hình bài toán tin học

- Biểu diễn các đối tượng thực tế: tìm cách tổ chức, xây dựng cấu trúc thích hợp để mô tả, phản ánh chính xác dữ liệu thực tế trên máy tính -> **xây dựng cấu trúc dữ liệu cho bài toán**
- Xây dựng các thao tác xử lý dữ liệu: thiết kế các **giải thuật** (thuật toán) tương ứng với cấu trúc dữ liệu đã chọn để thực hiện các yêu cầu xử lý trong thực tế
- Giải thuật phụ thuộc vào cấu trúc dữ liệu được chọn: giải thuật sẽ khả thi và thao tác xử lý sẽ hiệu quả nếu sử dụng cấu trúc dữ liệu phù hợp

3

Vai trò của CTDL

♦ Công thức của Niklaus Wirth

Data Structures + Algorithms = Programs
(Cấu trúc dữ liệu + Giải thuật = Chương trình)

4

Nội dung

♦ Vai trò của CTDL

♦ Tiêu chuẩn đánh giá CTDL

♦ Kiểu dữ liệu

♦ Con trỏ

♦ Mảng

♦ Chuỗi

♦ Cấu trúc

5

Tiêu chuẩn đánh giá CTDL

♦ Phản ánh đúng thực tế

- Cấu trúc dữ liệu quyết định tính đúng đắn của bài toán
- Đảm bảo các trạng thái biến đổi, miền giá trị của dữ liệu

♦ Phù hợp với các thao tác xử lý

- Tổ chức tốt dữ liệu -> giải thuật xử lý hiệu quả

♦ Tiết kiệm tài nguyên hệ thống

- Xác định tiêu chí lựa chọn tài nguyên ưu tiên: bộ nhớ, CPU

6

Nội dung

- ◆ Vai trò của CTDL
- ◆ Tiêu chuẩn đánh giá CTDL
- ◆ Kiểu dữ liệu
- ◆ Con trỏ
- ◆ Mảng
- ◆ Chuỗi
- ◆ Cấu trúc

7

Kiểu dữ liệu

- ◆ Kiểu dữ liệu nhằm phân loại dữ liệu, xác định miền giá trị của dữ liệu được lưu trữ và các thao tác có thể thực hiện trên tập giá trị của dữ liệu
- ◆ Các thuộc tính của kiểu dữ liệu
 - Tên kiểu dữ liệu
 - Miền giá trị
 - Kích thước lưu trữ
 - Tập các toán tử

8

Kiểu dữ liệu

- ◆ Các kiểu dữ liệu cơ sở
 - Kiểu có thứ tự rời rạc: số nguyên, kí tự, logic, liệt kê, ...
 - Kiểu liên tục: số thực
- ◆ Các kiểu dữ liệu có cấu trúc
 - Được tổ chức, liên kết các thành phần từ những kiểu dữ liệu cơ sở đã được định nghĩa
 - Kiểu dữ liệu có cấu trúc: mảng, chuỗi, tập tin, cấu trúc, ...

9

Nội dung

- ◆ Vai trò của CTDL
- ◆ Tiêu chuẩn đánh giá CTDL
- ◆ Kiểu dữ liệu
- ◆ Con trỏ
- ◆ Mảng
- ◆ Chuỗi
- ◆ Cấu trúc

10

Con trỏ

- ◆ Là một loại biến đặc biệt lưu trữ địa chỉ của một vùng nhớ khác (trỏ đến hoặc quản lý vùng nhớ đó)
- ◆ Được dùng để cấp phát và quản lý một hoặc một dãy các vùng nhớ liên tiếp
- ◆ Cho phép cấp phát động các vùng nhớ tùy theo nhu cầu sử dụng để tránh lãng phí
- ◆ Liên kết các vùng nhớ bị phân mảnh nhằm tiết kiệm bộ nhớ (danh sách liên kết)

11

Con trỏ

◆ Khai báo:

```
datatype *var;
```

- var là tên con trỏ có kiểu dữ liệu datatype

```
int *ptr;
```

```
char *s;
```

- Con trỏ var quản lý vùng nhớ có cùng kiểu dữ liệu

```
int a = 10;
```

```
ptr = &a;
```

12

Con trỏ

- ◆ Toán tử * cho phép truy xuất nội dung vùng nhớ mà con trỏ đang quản lý

```
int a = 10;
int *ptr = &a;
cout<<*ptr;
```

- ◆ Toán tử * và & sẽ loại trừ nhau theo thứ tự này

```
cout<<*(&a);
```

13

Đặc điểm con trỏ

- ◆ Con trỏ phải được khởi tạo (chỉ định vùng nhớ quản lý) trước khi truy xuất

```
int *ptr;
cout<<ptr; //runtime error
*ptr = 10; //runtime error
int a = 20;
ptr = &a;
cout<<*ptr;
```

14

Đặc điểm con trỏ

- ◆ Nếu vùng nhớ được quản lý bị thay đổi, tất cả con trỏ liên quan cũng ảnh hưởng theo

```
int a = 20;
int *p = &a;
a = a + 10;
int *q = p;
*q = *q - 5;
cout<<"*p = "<<*p<<"*q = "<<*q;
a = a*2;
cout<<"*p = "<<*p<<"*q = "<<*q;
```



15

Đặc điểm con trỏ

- ◆ Phép `+`, `-` là phép dịch chuyển địa chỉ vùng nhớ theo kích thước kiểu dữ liệu của con trỏ



```
int a = 20;
int *p = &a;
cout<<p<<" , "<<p+1;
```

16

Đặc điểm con trỏ

- ◆ Con trỏ không định kiểu (kiểu dữ liệu **void**) có thể trỏ đến bất kì vùng nhớ nào

```
int i = 20, *q;
double d = 15.5;
void *p;
q = &i;
p = &i;
p = &d;
```

- ◆ Con trỏ **NULL** không trỏ đến vùng nhớ nào

```
q = NULL;
```

17

Nội dung

- ◆ Vai trò của CTDL
- ◆ Tiêu chuẩn đánh giá CTDL
- ◆ Kiểu dữ liệu
- ◆ Con trỏ
- ◆ Mảng
- ◆ Chuỗi
- ◆ Cấu trúc

18

Kiểu mảng

- ◆ Là tập hữu hạn các dữ liệu (phần tử) có cùng kiểu
- ◆ Được sử dụng khi cần lưu trữ nhiều dữ liệu có cùng cấu trúc
- ◆ Các phần tử được lưu trữ trong các vùng nhớ liên tiếp nhau và được xác định thông qua chỉ số
- ◆ Mảng hai chiều có thể được xem là mảng một chiều, với mỗi phần tử là một mảng một chiều

19

Kiểu mảng

◆ Khai báo

- Mảng 1 chiều

```
KiểuDữLiệu TênMảng[kích_thước];
int iA[100];
double dA[200];
```

- Mảng 2 chiều

```
KiểuDữLiệu TênMảng[số_dòng][số_cột];
int iB[3][4];
double dB[50][40];
```

20

Kiểu mảng

◆ Truy xuất phần tử mảng

- Mảng 1 chiều: truy xuất thông qua tên mảng và chỉ số phần tử trong mảng

```
tên_mảng[chỉ_số];
cin>>iA[0];
```

- Mảng 2 chiều: truy xuất thông qua tên mảng, chỉ số dòng và chỉ số cột

```
tên_mảng[chỉ_số_dòng][chỉ_số_cột];
cin>>iB[0][3];
```

21

Kiểu mảng

- ◆ Khi truyền mảng 1 chiều là tham số cho hàm thì không chỉ định kích thước khi khai báo

```
void NhapMang(int A[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cout<<"A["<<i<<" = ";
        cin>>A[i];
    }
}

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

22

Kiểu mảng

- ◆ Khi truyền mảng 2 chiều là tham số cho hàm thì không chỉ định kích thước dòng khi khai báo

```
void NhapMang(int A[][100], int m, int n)
{
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
        {
            cout<<"A["<<i<<"]["<<j<<" = ";
            cin>>A[i][j];
        }
}
```

23

Con trỏ và mảng

- ◆ Địa chỉ phần tử đầu tiên cũng là địa chỉ của mảng

```
//A và &A[0] có cùng địa chỉ
int A[100], *pA;
NhapMang(A, n);
cout<<"địa chỉ của A:"<<A;
cout<<"địa chỉ A[0]:"<<&A[0];
pA = &A[0]; //hoặc lệnh pA = A
```

- ◆ Mảng là con trỏ hằng (địa chỉ cố định)

```
A = pA; //thao tác không hợp lệ
```

24

Con trỏ và mảng

◆ Truy xuất phần tử mảng dưới góc độ con trỏ

```
voidNhapMang(int A[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cout<<"A["<<i<<" ] = ";
        cin>>A + i;
    }
}

voidXuatMang(int A[], int n)
{
    for (int i = 0; i < n; i++)
        cout<<*(A + i)<<" ";
}
```

25

Nội dung

- ◆ Vai trò của CTDL
- ◆ Tiêu chuẩn đánh giá CTDL
- ◆ Kiểu dữ liệu
- ◆ Con trỏ
- ◆ Mảng
- ◆ Chuỗi
- ◆ Cấu trúc

26

Chuỗi ký tự

- ◆ Dãy hữu hạn các ký tự và được kết thúc bằng ký tự '\0' (NULL)
- ◆ Chuỗi ký tự được đặt trong dấu nháy kép
"cau truc du lieu va giai thuat"
- ◆ Các thao tác xử lý chuỗi là các hàm thuộc thư viện <string.h>

27

Chuỗi ký tự

◆ Khai báo chuỗi

```
char string_var[size]; hoặc char *string;
char ho_ten[30], *s;
```

- Khi khai báo kiểu con trỏ phải chú ý đến vấn đề cấp phát vùng nhớ

```
char *s = new char[100];
```

- Khai báo và khởi tạo chuỗi:

```
char *s = "cong nghe thong tin";
```

28

Chuỗi ký tự

◆ Một số hàm xử lý chuỗi

Tên hàm	Chức năng văn tắt
gets(s)	Nhập chuỗi có chứa khoảng trắng
puts(s)	Xuất chuỗi ra màn hình
strlen(s)	Tính độ dài của chuỗi
strcpy(src, des)	Copy des vào src
strcat(s1, s2)	Ghép chuỗi s2 vào sau chuỗi s1
strupr(s)	Chuyển tất cả ký tự của s thành viết hoa
strlwr(s)	Chuyển tất cả ký tự của s thành viết thường
strrev(s)	Đảo ngược chuỗi s
strcmp(s1, s2)	Số sánh 2 chuỗi s1 và s2
strstr(text, pat)	Tìm kiếm vị trí xuất hiện đầu tiên của pat trong text.

29

Nội dung

- ◆ Vai trò của CTDL
- ◆ Tiêu chuẩn đánh giá CTDL
- ◆ Kiểu dữ liệu
- ◆ Con trỏ
- ◆ Mảng
- ◆ Chuỗi
- ◆ Cấu trúc

30

Kiểu cấu trúc

◆ Khái niệm

- Là kiểu dữ liệu gồm nhiều thành phần, các thành phần có kiểu dữ liệu khác nhau

◆ Thành phần của cấu trúc được gọi là trường (field)

◆ Kiểu cấu trúc đóng gói các dữ liệu khác nhau và tham chiếu chúng như một thực thể

◆ Kiểu cấu trúc còn được gọi là kiểu dữ liệu tự định nghĩa (user defined type)

31

Định nghĩa kiểu cấu trúc

◆ Cú pháp

```
struct <TênCấuTrúc>
{
    <Kiểu 1> <field 1>;
    <Kiểu 2> <field 2>;
    ...
    <Kiểu n> <field n>;
};
```

◆ Sau khi được định nghĩa, <TênCấuTrúc> được xem như là một kiểu dữ liệu mới

Company Logo

32

Định nghĩa kiểu cấu trúc

◆ Định nghĩa kiểu mới từ các kiểu dữ liệu cơ sở

```
struct Diem
{
    double x, y;
};

struct NgayThang
{
    int ng, th, nam;
};
```

◆ Diem và NgayThang bây giờ có thể được xem như một kiểu dữ liệu mới, được lập trình viên tự định nghĩa

```
Diem A, B, C;
Diem P[100];
```

Company Logo

33

Sử dụng kiểu cấu trúc

- ◆ Sử dụng toán tử `.` để truy xuất đến các thành phần của biến cấu trúc

- ◆ Cú pháp: `<TênBiến>.<TênTrường>`

```
Diem A, B;
double AB;

A.x = 2;
A.y = 5;
B.x = 8;
B.y = 7;

AB = sqrt(sqr(B.x - A.x) + sqr(B.y - A.y));
```

34

Sử dụng kiểu cấu trúc

- ◆ Các biến có cùng kiểu cấu trúc có thể được gán cho nhau

```
Diem A, B;
A.x = 10;
A.y = 4;
B = A; //tương đương B.x = A.x; B.y = A.y
```

35

Sử dụng kiểu cấu trúc

- ◆ Các hàm nhập/xuất chuẩn không áp dụng cho biến cấu trúc

```
SinhVien s;
cin>>s; //không hợp lệ
cout<<s; //không hợp lệ
strcpy(s.MSSV, "091234");
strcpy(s.HoTen, "Le Van Nghia");
cin>>s.NTNS.ng;
cin>>s.NTNS.th;
cin>>s.NTNS.nam;
cin>>s.GioiTinh;
```

```
struct SinhVien
{
    char MSSV[50];
    char HoTen[100];
    NgayThang NTNS;
};
```

36

Sử dụng kiểu cấu trúc

- ◆ Không thể áp dụng các phép toán số học (+, -, *, /, ...), phép toán so sánh cho biến có kiểu cấu trúc

```
Diem a, b;
...
if (a < b) //không hợp lệ
    cout<<"2 diem trung nhau";
else cout<<"2 diem khác nhau";
```

37

Sử dụng kiểu cấu trúc

- ◆ Khởi tạo giá trị cho biến cấu trúc tại thời điểm khai báo

- Giá trị các trường được đặt giữa cặp dấu { và }, chúng được phân cách nhau bởi dấu phẩy

```
Diem A = {5,8}, B = {4,7}, C = {8,2};
TamGiac t = {A, B, C};
```

```
struct TamGiac
{
    Diem A, B, C;
};
```

38

Sử dụng kiểu cấu trúc

- ◆ Khởi tạo giá trị cho biến cấu trúc có thành phần cũng là một kiểu cấu trúc

```
struct NhanVien
{
    char Ten[50];
    NgayThang NTNS;
    double BacLuong;
};
```

```
NhanVien nv = {"Hung", {5, 7, 1975}, 2.77};
```

39

Cấu trúc và con trỏ

- ◆ Các thành phần của một biến kiểu cấu trúc chiếm một vùng nhớ cụ thể trên RAM

- ◆ Địa chỉ của biến có kiểu cấu trúc là địa chỉ của thành phần đầu tiên

```
void main()
{
    Diem A, B, C;
    ...
    TamGiac t = {A, B, C};
    //&t == &t.A == &t.A.x
    cout<<&t<<endl;
    cout<<&t.A <<endl;
    cout<<&t.A.x;
    ...
}
```

40

Cấu trúc và con trỏ

- ◆ Con trỏ cấu trúc cần phải được cấp phát vùng nhớ quản lý trước khi truy xuất

```
struct Diem
{
    double x, y;
};

void main()
{
    Diem *pA;
    //cấp phát vùng nhớ quản lý
    pA = new Diem;
}
```

41

Cấu trúc và con trỏ

- ◆ Con trỏ cấu trúc cần phải được chỉ định vùng nhớ quản lý trước khi truy xuất.

```
struct Diem
{
    double x, y;
};

void main()
{
    Diem A = {3, 8};
    Diem *pA;
    //chỉ định vùng nhớ cho con trỏ pA quản lý
    pA = &A;
}
```

42

Cấu trúc và con trỏ

◆ Truy xuất thành phần của con trỏ cấu trúc

- Sử dụng toán tử `->` giữa biến con trỏ và biến thành phần

`<têncontrỏ> -> <tênthànhphần>`
`cin>>pA->x; cout<<pA->x;`

- Sử dụng toán tử chấm . truy xuất đến nội dung vùng nhớ mà con trỏ quản lý

`(*<têncontrỏ>).<tênthànhphần>`
`cin>>(*pA).x; cout<<(*pA).x;`

43
