

Mục tiêu

Nắm được các kiến thức cơ bản về con trỏ (pointer).

Nội dung

Con trỏ

Cần hiểu rõ: giá trị của biến con trỏ là địa chỉ vùng nhớ của biến khác hoặc vùng nhớ hợp lệ đã được cấp phát.

Cách khai báo:

```
/// khai báo biến con trỏ kiểu int
int *p;
int a = 10;

/// gán giá trị biến con trỏ là địa chỉ biến khác
p = &a;

/// cấp phát vùng nhớ riêng và sử dụng
p = (int*)malloc(sizeof(int));
/// hoặc sử dụng toán tử new
p = new int;

if (p != NULL)
{
    *p = 100;
}
```

Phép toán đối với con trỏ: chủ yếu phép + và -, trong đó về bản chất là thay đổi địa chỉ lưu bởi con trỏ tương ứng với kiểu dữ liệu (nghĩa là dịch chuyển địa chỉ đi 1 khoảng tương ứng với số byte của kiểu dữ liệu).

Xét ví dụ sau:

```
///khai báo 2 con trỏ p kiểu int (4 byte) và q kiểu char (1 byte)
int *p;
char *q;
int a = 10;

///gán giá trị cho cả 2 con trỏ là địa chỉ của biến a
p = &a;
q = (char*)&a;

///vì cùng giá trị địa chỉ (là biến a) nên hiệu bằng 0
int hieu = (int)p - (int)q;

///khi cộng thêm 1 đơn vị thì hiệu không là 0 mà là 3
///lý do: kiểu int 4 byte nên p + 1 sẽ tăng giá trị địa chỉ lên 4,
```

Thực hành Kỹ thuật lập trình

```
//còn kiểu char chỉ 1 byte nên q + 1 chỉ tăng giá trị địa chỉ lên 1
hieu = (int)(p + 1) - (int)(q + 1);
```

Sử dụng con trỏ với mảng tĩnh: dựa vào ý nghĩa phép toán của con trỏ ở trên ứng dụng con trỏ để duyệt mảng.

```
//khai báo mảng tĩnh
int mang[10];
//khai báo con trỏ lưu giá trị là địa chỉ đầu mảng
int *p = mang;
//duyet mảng bình thường dựa vào con trỏ để gán giá trị mảng về 0
for (int i=0; i<10; ++i)
{
    *(p+i) = 0;
    //tương đương a[i] = 0
}
```

Sử dụng con trỏ với mảng cấp phát động: không như mảng tĩnh, phải khai báo số lượng phần tử tối đa ngay từ đầu, với con trỏ thì cho phép xin cấp phát đúng số lượng vùng nhớ cần thiết.

```
/// khai báo con trỏ là mảng động
int n = 10;
int *p;

/// sử dụng malloc
p = (int*)malloc(sizeof(int) * n);
/// hoặc sử dụng calloc
p = (int*)calloc(n, sizeof(int));
/// hoặc sử dụng toán tử new
p = new int[n];

if (p == NULL)
{
    return;
}
/// sau đó sử dụng như mảng thường
for (int i = 0; i < n; ++i)
{
    /// 2 dòng code là tương đương
    *(p + i) = 0;
    p[i] = 0;
}
/// sau khi sử dụng con trỏ thì phải nhớ thu hồi vùng nhớ
free(p);    /// nếu sử dụng hàm cấp phát malloc/calloc
delete[] p; /// nếu sử dụng toán tử new
```

Kỹ thuật phát sinh số ngẫu nhiên

Để phát sinh số ngẫu nhiên cần sử dụng 2 thư viện <time.h> (để lấy số thời gian hiện tại) và thư viện <stdlib.h> (chứa hàm phát sinh số ngẫu nhiên).

Thực hành Kỹ thuật lập trình

Trước khi sử dụng được hàm phát sinh số ngẫu nhiên `rand()` thì trước hết phải thực hiện gieo hạt giống cho nó thông qua hàm `srand(unsigned int)`, và chỉ cần gieo 1 lần đầu cho suốt chương trình là được.

Vấn đề cần có giá trị trong khoảng $[a, b]$ mong muốn thì sử dụng phép đồng dư (%).

```
//cần có dòng lệnh này trước khi gọi hàm rand() lần đầu tiên
srand((unsigned)time(NULL));
//giá trị lấy được sẽ nằm trong khoảng [a, b]
int x = rand() % (b - a + 1) + a;
```

Con trỏ và struct

Như đã học ở tuần trước thì struct thực ra là kiểu dữ liệu do người dùng tự định nghĩa. Vì vậy sử dụng con trỏ với struct là hoàn toàn tương tự kiểu dữ liệu nguyên thủy (int, long, double,...)

Ví dụ cơ bản

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

struct PhanSo
{
    int tu, mau;
};

void main()
{
    PhanSo *pPS = (PhanSo*)malloc(sizeof(PhanSo));
    if (pPS == NULL)
    {
        return;
    }
    pPS->tu = 9;
    pPS->mau = 10;
    printf("%d/%d", pPS->tu, pPS->mau);

    free(pPS);

    getch();
}
```

Với con trỏ thì luôn phải nhớ cấp phát trước khi sử dụng và thu hồi khi không dùng nữa.

Con trỏ trong truyền tham số

Vì giá trị của biến con trỏ là địa chỉ vùng nhớ nên khi truyền tham số là con trỏ thì nghĩa là truyền địa chỉ vùng nhớ (tương tự truyền tham số kiểu tham chiếu với toán tử **&**).

Xét ví dụ sau:

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

struct PhanSo
{
    int tu, mau;
};

void DaoPhanSo(PhanSo ps)
{
    int temp = ps.mau;
    ps.mau = ps.tu;
    ps.tu = temp;
}

void DaoPhanSoPointer(PhanSo *pPS)
{
    int temp = pPS->mau;
    pPS->mau = pPS->tu;
    pPS->tu = temp;
}

void main()
{
    PhanSo *pPS = (PhanSo*)malloc(sizeof(PhanSo));
    if (pPS == NULL)
    {
        return;
    }
    pPS->tu = 9;
    pPS->mau = 10;
    printf("Ket qua sau khi gọi ham DaoPhanSo(PhanSo ps)\n");
    DaoPhanSo(*pPS);
    printf("%d/%d\n", pPS->tu, pPS->mau);
    printf("Ket qua sau khi gọi ham DaoPhanSoPointer(PhanSo *pPS)\n");
    DaoPhanSoPointer(pPS);
    printf("%d/%d\n", pPS->tu, pPS->mau);

    free(pPS);

    getch();
}
```

Lưu ý toán tử * để lấy giá trị tại địa chỉ con trỏ lưu giữ (sử dụng khi cần truyền tham số kiểu tham trị đối với con trỏ như ví dụ trên).

Bài tập

Bài 1

Thực hiện các yêu cầu sau với con trỏ:

- Viết hàm phát sinh mảng (số lượng phần tử ít nhất là 3, giá trị mỗi phần tử nằm trong khoảng cho trước).

Thực hành Kỹ thuật lập trình

- Tìm phần tử lớn nhất và nhỏ nhất của mảng.
- Tính tổng tất cả các phần tử của mảng.
- Tính tổng tất cả các phần tử có số lần xuất hiện lớn nhất.
- Tìm phần tử trung vị của mảng.
- Tìm mảng con của mảng (xác định 1 mảng có là con của mảng đã có hay không).

Bài 2

Viết chương trình phát sinh mảng với con trỏ (số lượng phần tử ít nhất là 10, giá trị mỗi phần tử nằm trong khoảng [-1000, 2000]). Thực hiện các yêu cầu sau:

- Tìm số chính phương đầu tiên trong mảng.
- Tìm mảng con là tất cả các số nguyên tố trong mảng.

Bài 3

Viết chương trình phát sinh 2 mảng động với con trỏ p, q (số lượng phần tử ít nhất 10, giá trị mỗi phần tử nằm trong khoảng [-50, 50]). Sau đó thực hiện các yêu cầu sau:

- Xuất ra màn hình các phần tử chỉ có trong mảng p mà không có trong mảng q.
- Tạo 1 mảng là hợp của 2 mảng p, q sao cho các phần tử khác nhau đôi một.

Bài 4

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. Áp dụng cho hàm main sau:

```
struct ThoiGian
{
    int gio, phut, giay;
};

ThoiGian* TaoThoiGian(int h, int m, int s)
{
    // ...
}

void HopLeThoiGian(ThoiGian *pTG)
{
    // ...
}

void PrintTG(ThoiGian *pTG)
{
    // ...
}

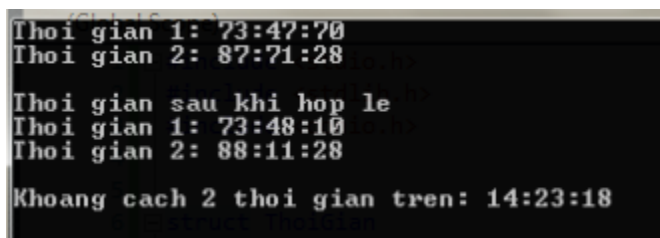
ThoiGian* KhoangCachThoiGian(ThoiGian *pTG1, ThoiGian *pTG2)
{
    // ...
}
```

```
}

void main()
{
    // khởi động random
    srand(3072);
    // khai báo con trỏ ThoiGian
    ThoiGian *pTG1, *pTG2;
    // tạo giá trị
    pTG1 = TaoThoiGian(rand()%100, rand()%100, rand()%100);
    pTG2 = TaoThoiGian(rand()%100, rand()%100, rand()%100);
    // xuất ra màn hình
    printf("Thoi gian 1: ");
    PrintTG(pTG1);
    printf("\n");
    printf("Thoi gian 2: ");
    PrintTG(pTG2);
    printf("\n\n");
    // làm hợp lệ thời gian
    HopLeThoiGian(pTG1);
    HopLeThoiGian(pTG2);
    printf("Thoi gian sau khi hop le\n");
    printf("Thoi gian 1: ");
    PrintTG(pTG1);
    printf("\n");
    printf("Thoi gian 2: ");
    PrintTG(pTG2);
    printf("\n\n");
    // tính khoảng cách
    ThoiGian *p = KhoangCachThoiGian(pTG1, pTG2);
    printf("Khoang cach 2 thoi gian tren: ");
    PrintTG(p);
    printf("\n");

    getch();
}
```

Kết quả



```
Thoi gian 1: 73:47:70
Thoi gian 2: 87:71:28

Thoi gian sau khi hop le
Thoi gian 1: 73:48:10
Thoi gian 2: 88:11:28

Khoang cach 2 thoi gian tren: 14:23:18
```

Bài 5

Viết chương trình sử dụng con trỏ cấu trúc thể hiện ngày, tháng, năm ra màn hình, và tính khoảng cách giữa 2 ngày.