



Con trỏ - Pointer

Con trỏ – Pointer

- Khai báo
- Các toán tử “&”, “*”, “=”, “+”
- Nhắc lại về truyền tham số địa chỉ
- Con trỏ và mảng
- Cấp phát vùng nhớ động

Con trỏ – Một số lý do nên sử dụng

- Con trỏ là kiểu dữ liệu lưu trữ địa chỉ của các vùng dữ liệu trong bộ nhớ máy tính
- Kiểu con trỏ cho phép:
 - Truyền tham số kiểu địa chỉ
 - Biểu diễn các kiểu, cấu trúc dữ liệu động
 - Lưu trữ dữ liệu trong vùng nhớ heap
- Con trỏ đã được sử dụng trong hàm scanf

Con trỏ – Khai báo trong C

Kiểu con trỏ phải được định nghĩa trên một kiểu cơ sở đã được định nghĩa trước đó.

```
typedef kiểu cơ sở *Tên kiểu;
```

```
typedef      int      *PINT;
```

//PINT là kiểu con trỏ - địa chỉ vùng nhớ kiểu int

```
int      x;
PINT    p; //p, p1: biến kiểu int *
int      *p1;
```

Con trỏ – Khai báo trong C

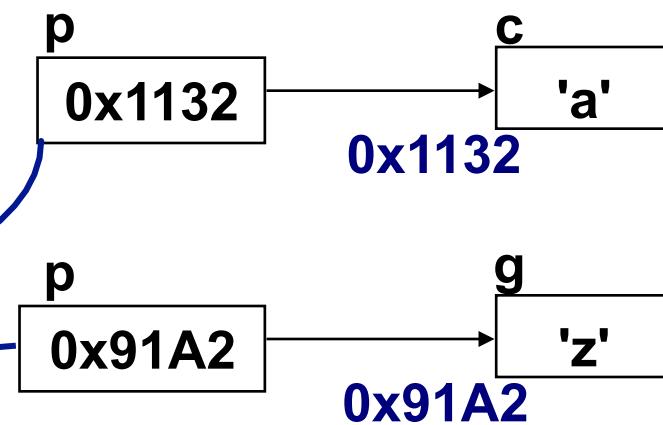
```
int      *pi;  
  
long int *p;  
  
float   *pf;  
  
char    c, d, *pc; /* c và d kiểu char  
                    pc là con trỏ đến char */  
  
double *pd, e, f; /* pd là con trỏ đến double  
                    e and f are double */  
  
char    *start, *end;
```

Con trỏ - Toán tử “&”

- “&”: toán tử lấy địa chỉ của 1 biến
- Địa chỉ của tất cả các biến trong chương trình đều đã được chỉ định từ khi khai báo*

```
char g = 'z';

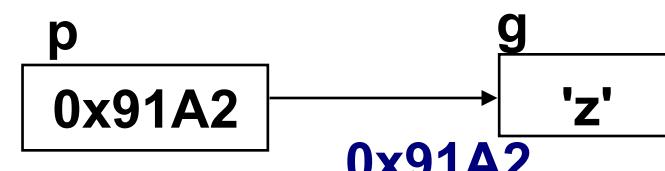
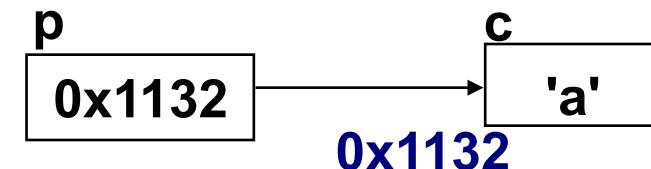
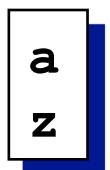
int main()
{
    char c = 'a';
    char *p;
    p = &c;
    p = &g;
    return 0;
}
```



Con trỏ - Toán tử “*”

- “*”: toán tử truy xuất giá trị của vùng nhớ được quản lý bởi con trỏ.

```
#include <stdio.h>
char g = 'z';
int main()
{
    char c = 'a';
    char *p;
    p = &c;
    printf("%c\n", *p);
    p = &g;
    printf("%c\n", *p);
    return 0;
}
```



xuất giá trị do p đang
quản lý

Con trỏ - Truyền tham số địa chỉ

```
#include <stdio.h>
void change(int *v);

int main()
{
    int var = 5;
    change(&var);
    printf("main: var = %i\n", var);
    return 0;
}

void change(int *v)
{
    (*v) *= 100;
    printf("change: *v = %i\n", (*v));
}
```

Con trỏ NULL

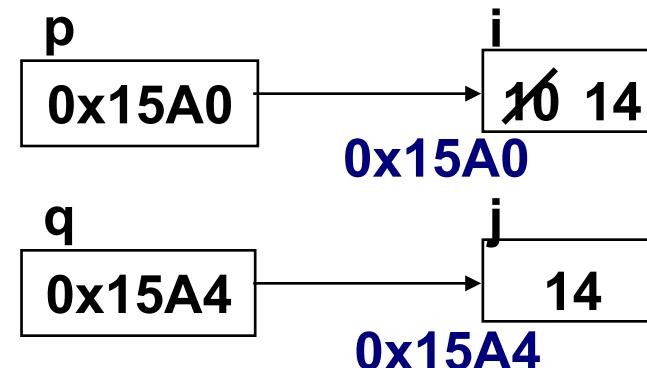
- Giá trị đặc biệt để chỉ rằng con trỏ không quản lý vùng nào. Giá trị này thường được dùng để chỉ một con trỏ không hợp lệ.

```
#include <stdio.h>
int main()
{
    int i = 13;
    short *p = NULL;
    if (p == NULL)
        printf("Con trỏ không hợp lệ!\n");
    else
        printf("Giá trị : %0xi\n", *p);
    return 0;
}
```

Con trỏ - Toán tử gán “=”

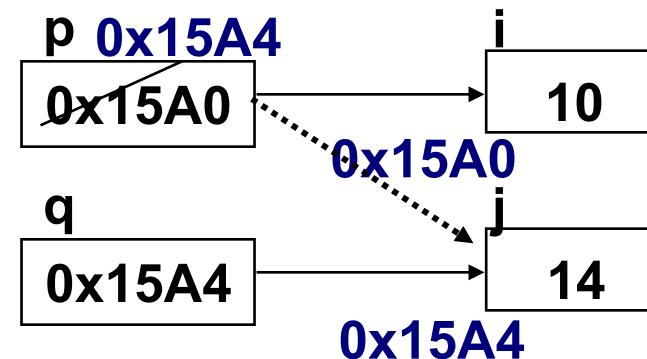
- Có sự khác biệt rất quan trọng khi thực hiện các phép gán:

```
int i = 10, j = 14;  
int *p = &i;  
int *q = &j;  
  
*p = *q;
```



và:

```
int i = 10, j = 14;  
int *p = &i;  
int *q = &j;  
  
p = q;
```



Luyện tập – Điện vào ô trống

```
int main(void)
{
    int i = 10, j = 14, k;
    int *p = &i;
    int *q = &j;

    *p += 1;
    p = &k;
    *p = *q;
    p = q;
    *p = *q;

    return 0;
}
```

i

0x2100

j

0x2104

k

0x1208

p

0x120B

q

0x1210

Con trỏ và Mảng

- Biến kiểu mảng là địa chỉ tĩnh của một vùng nhớ, được xác định khi khai báo, không thay đổi trong suốt chu kỳ sống.
- Biến con trỏ là địa chỉ động của một vùng nhớ, được xác định qua phép gán địa chỉ khi chương trình thực thi.

```
#include <stdio.h>
int main()
{
    int a[10] = {1, 3, 4, 2, 0};
    int *p;
    p = a; //a = p: sai
    printf("0x%08X %i 0x%08X %i\n",
           a, a[0], p, *p);
    return 0;
}
```

Con trỏ - Toán tử “+” với số nguyên

```
#include <stdio.h>
int main()
{
    short a[10] = {1, 3, 5, 2, 0};
    short *p = a;
    printf("0x%08X %i 0x%08X %i\n", );
        a, a[0], p, *p);
    p++;
    printf("0x%08X %i 0x%08X %i\n", );
        a, a[0], p, *p);
    (*p)++;
    printf("0x%08X %i 0x%08X %i\n", );
        a, a[0], p, *p);
    return 0;
}
```

0x15A0

a
1
4
5
2
0
...

0x16B2

p
0x15A0

Con trỏ - Luyện tập

```
#include <stdio.h>
int main()
{
    int a[10] = {2, 3, 5, 1, 4, 7, 0};
    int *p = a;
    printf("%i %i\n", a[0], *p);
    p++;
    printf("%i %i\n", *p, p[2]);
    p++; a[2] = 9;
    printf("%i %i\n", p[1], *p);
    p -= 2;
    printf("%i %i\n", p[3], p[1]);
    return 0;
}
```

2	2
3	1
1	9
1	3

Con trỏ - Cấp phát vùng nhớ động

- Có thể chỉ định vùng mới cho 1 con trỏ quản lý bằng các lệnh hàm **malloc** (memory-alloc), **calloc** (clear-alloc) hoặc toán tử **new** của C++
- Vùng nhớ do lập trình viên chỉ định phải được giải phóng bằng lệnh **free** (malloc, calloc) hoặc toán tử **delete** (new)

```
#include <stdio.h>
int main()
{
    int *p = malloc(10*sizeof(int));
    p[0] = 1;
    p[3] = -7;
    free(p);
    return 0;
}
```

Tổng kết

- Khai báo
- Các toán tử “&”, “*”, “=”, “+”
- Nhắc lại về truyền tham số địa chỉ
- Con trỏ và mảng
- Cấp phát vùng nhớ động



Chuỗi ký tự - String

Chuỗi ký tự – Strings

- Một số qui tắc
- Nhập / xuất
- Con trỏ và chuỗi ký tự
- Một số hàm thư viện

Chuỗi ký tự - Một số qui tắc

- Chuỗi ký tự là mảng một chiều có mỗi thành phần là một số nguyên được kết thúc bởi số 0.
- Ký tự kết thúc (0) ở cuối chuỗi ký tự thường được gọi là ký tự **null** (không giống con trỏ NULL). Có thể ghi là 0 hoặc '\0' (không phải chữ o).
- Được khai báo và truyền tham số như mảng một chiều.

```
char          s [100] ;  
unsigned char s1 [1000] ;
```

Chuỗi ký tự - Ví dụ

```
char first_name[5] = { 'J', 'o', 'h', 'n', '\0' };  
  
char last_name[6] = "Minor";  
  
char other[] = "Tony Blurt";  
  
char characters[7] = "No null";
```

first_name	'J'	'o'	'h'	'n'	0						
last_name	'M'	'i'	'n'	'o'	'r'	0					
other	'T'	'o'	'n'	'y'	32	'B'	'T'	'u'	'r'	't'	0
characters	'N'	'o'	32	'n'	'u'	'T'	'T'	'T'	'T'	'T'	0

Chuỗi ký tự - Nhập / xuất

- Có thể nhập / xuất chuỗi ký tự s bằng cách nhập từng ký tự của s
- Hoặc sử dụng các hàm scanf và printf với ký tự định dạng "%s"

```
char other[] = "Tony Blurt";
printf("%s\n", other);
```
- Nhập chuỗi có khoảng trắng dùng hàm gets

```
char name[100];
printf("Nhập một chuỗi ký tự %s: ");
gets(name);
```

- scanf and gets have a potential overflow problem (use fgets(s, len, stdin) instead to specify the maximum buffer)

Lưu ý: kết thúc chuỗi

```
#include <stdio.h>

int main()
{
    char other[] = "Tony Blurt";

    printf("%s\n", other);

    other[4] = '\0';

    printf("%s\n", other);

    return 0;
}
```

"Blurt" sẽ không
được in ra

Tony Blurt
Tony

other	'T'	'o'	'n'	'y'	32	'B'	'l'	'u'	'r'	't'	0
-------	-----	-----	-----	-----	----	-----	-----	-----	-----	-----	---

Chuỗi ký tự – Một số hàm thư viện

- Lấy độ dài chuỗi

l = strlen(s) ;

- Đổi toàn bộ các ký tự của chuỗi thành IN HOA

strupr(s) ;

- Đổi toàn bộ các ký tự của chuỗi thành in thường

strlwr(s) ;

Chuỗi ký tự – Một số hàm thư viện

- So sánh chuỗi: so sánh theo thứ tự từ điển

Phân biệt IN HOA – in thường:

```
int strcmp(const char *s1, const char *s2);
```

Không phân biệt IN HOA – in thường:

```
int stricmp(const char *s1, const char *s2);
```

Chuỗi ký tự – ví dụ strcmp

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

int main()
{
    char s1[] = "Minor";
    char s2[] = "Tony";
    int cmp = strcmp(s1, s2);
    if (cmp < 0)
        printf("%s < %s", s1, s2);
    else
        if (cmp == 0)
            printf("%s = %s", s1, s2);
        else
            printf("%s > %s", s1, s2);
    return 0;
}
```

Minor < Tony

Chuỗi ký tự – Một số hàm thư viện

- Gán nội dung chuỗi:
 - Chép toàn bộ chuỗi source sang chuỗi dest:

```
int strcpy(char *dest, const char *src);
```
 - Chép tối đa n ký tự từ source sang dest:

```
int strncpy(char *dest,  
            const char *src, int n);
```
- Tạo chuỗi mới từ chuỗi đã có:

```
char *strdup(const char *src);
```

Chuỗi ký tự – ví dụ strcpy

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

int main()
{
    char s[] = "Tony Blurt";
    char s2[100], *s3;

    strcpy(s2, s);
    printf("%s\n", s2);
    strncpy(s2 + 2, "12345", 3);
    printf("%s\n", s2);
    s3 = strdup(s + 5);
    printf("%s\n", s3);
    free(s3);
    return 0;
}
```

Tony Blurt
To123Blurt
Blurt

Chuỗi ký tự – Một số hàm thư viện

- Nối chuỗi:

```
char *strcat(char *dest,  
                const char *src) ;
```

- Tách chuỗi:

```
char *strtok(char *s,  
                const char *sep) ;
```

*Trả về địa chỉ của đoạn đầu tiên. Muốn tách đoạn kế tiếp
tham số thứ nhất sẽ là NULL*

Chuỗi ký tự – ví dụ strtok

```
#include <stdio.h>
#include <string.h>
#define SEPARATOR ". , "

int main()
{
    char s[] = "Thu strtok: 9,123.45";
    char *p;

    p = strtok(s, SEPARATOR);
    while (p != NULL)
    {
        printf("%s\n", p);
        p = strtok(NULL, SEPARATOR);
    }
    return 0;
}
```

Thu
strtok:
9
123
45

Chuỗi ký tự – Một số hàm thư viện

- Tìm một ký tự trên chuỗi:

```
char *strchr(const char *s, int c);
```

- Tìm một đoạn ký tự trên chuỗi:

```
char *strstr(const char *s1,  
             const char *s2);
```

Chuỗi ký tự – ví dụ tìm kiếm

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

int main()
{
    char s[] = "Thu tim kiem chuoi";
    char *p;

    p = strchr(s, 'm');
    printf("%s\n", p);
    p = strstr(s, "em");
    printf("%s\n", p);
    return 0;
}
```

m kiem chuoi
em chuoi

Chuỗi ký tự – chèn một đoạn ký tự

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

void StrIns(char *s, char *sub)
{
    int len = strlen(sub);
    memmove(s + len, s, strlen(s)+1);
    strncpy(s, sub, len);
}

int main()
{
    char s[] = "Thu chen";
    StrIns(s, "123");
    StrIns(s + 8, "45");
    printf("%s\n", s);
    printf("%s\n", p);
    return 0;
}
```

123 Thu chen
123 Thu 45chen

Chuỗi ký tự – xóa một đoạn ký tự

```
#include <stdio.h>

void StrDel(char *s, int n)
{
    memmove(s, s + n, strlen(s+n)+1);
}
int main()
{
    char s[] = "Thu xoa 12345";
    StrDel(s, 4);        printf("%s\n", s);
    StrDel(s + 4, 3);   printf("%s\n", p);
    return 0;
}
```

xoa 12345
xoa 45

Tổng kết

- Khai báo
- Nhập / xuất
- Con trỏ và chuỗi ký tự
- Một số hàm thư viện
- Chèn / loại bỏ một đoạn con