Bài giảng trên lớp

Bài giảng trên lớp là tài liệu gợi ý và hỗ trợ giảng viên trong quá trình lên lớp, tuy nhiên giảng viên cần có sự chuẩn bị của riêng mình cho phù hợp với từng lớp học.

Nếu giảng viên thiết kế bài giảng tốt hơn tài liệu đã cung cấp, xin hãy chủ động làm và gửi lại cho chúng tôi.

Nếu giảng viên cần thay đổi tài liệu đã cung cấp, những thay đổi có liên quan đến cấu trúc, nội dung kiến thức và có tính đổi mới, xin hãy chủ động làm và gửi lại cho chúng tôi.

Mọi ý kiến xin gửi cho Phòng NC-PTCT để được xem xét và ban hành chính thức và góp phần ngày càng hoàn thiện hơn học liệu của trường.

Trân trọng cảm ơn.

Liên hệ: Phòng NC-PTCT – FPT Polytechnic

Những ý tưởng đổi mới sẽ được xem xét và gửi qua Dự án CNGD để có sự hỗ trợ giảng viên làm nghiên cứu khoa học, hoặc hướng dẫn viết bài báo đăng trên tạp chí CNGD.

Lập trình C++
Bài 5 Con trỏ và tham chiếu

Hệ thống bài cũ

- Khai báo, khởi tạo, và truy xuất mảng một chiều.
- Xử lý mảng bằng vòng lặp for.
- Truyền mảng đến hàm.
- Cơ chế typedef.
- Một số phương pháp sắp xếp, tìm kiếm trên mảng.
- Mảng đa chiều.
- Xâu ký tự chuẩn.
- Tệp (file).

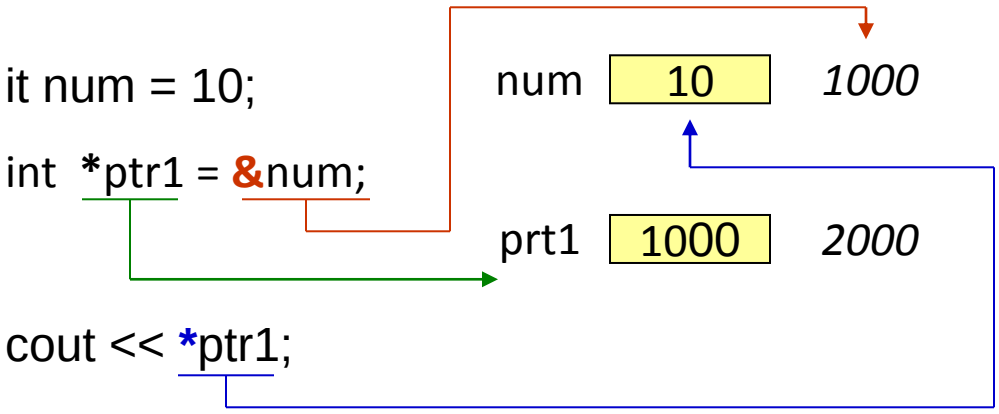
NỘI DUNG

- Con trỏ và tham chiếu
- Con trỏ hằng
- Con trỏ hàm
- Mảng và con trỏ

Con trỏ

Con trỏ đơn giản chỉ là *địa chỉ* của một vị trí bộ nhớ và cung cấp cách gián tiếp để truy xuất dữ liệu trong bộ nhớ

Ví dụ:

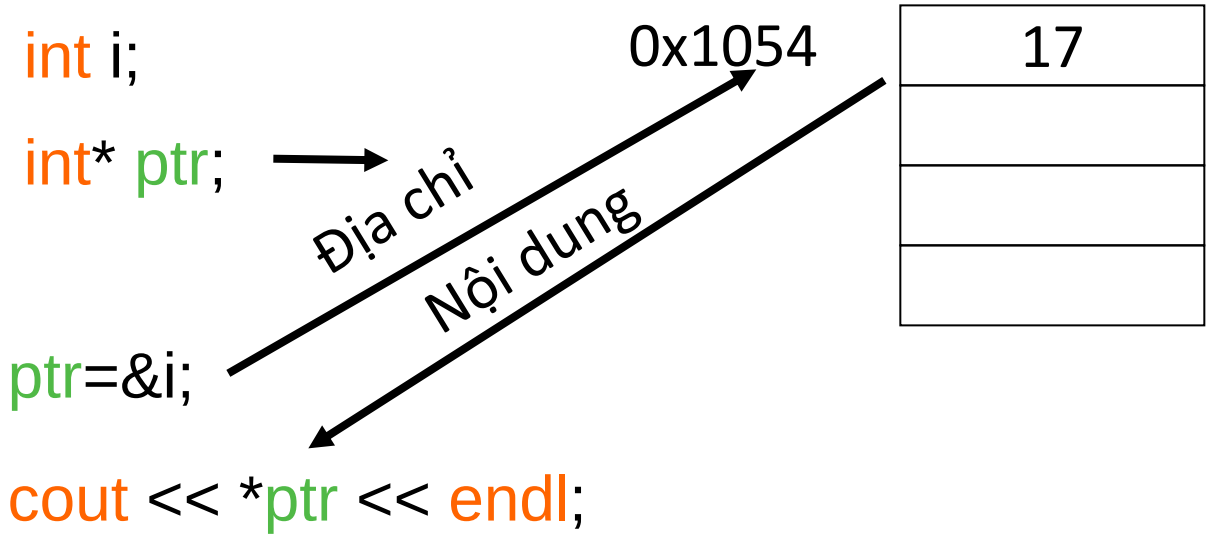


Biến con trỏ

- Biến con trỏ là biến lưu giá trị của địa chỉ vùng nhớ.
- Mỗi kiểu dữ liệu có một biến con trỏ riêng: con trỏ kiểu int, con trỏ kiểu char…
- C++ sử dụng:
  - Toán tử tham chiếu (&) để lấy địa chỉ của biến
  - Toán tử tham chiếu ngược (*)để lấy nội dung của biến được trỏ.
- Ví dụ:

```
int i=17;
int* ptr;   // khai báo biến trỏ kiểu int
ptr= &i;  // gán địa chỉ của biến i cho con trỏ ptr
cout << *ptr << endl;  // hiển thị nội dung của biến i
```

Biến con trỏ ...



Biến con trỏ ...

```
int v;        // khai báo biến v kiểu int
int w;        // khai báo biến w kiểu int
int* p;      // khai báo biến p kiểu con trỏ trỏ tới kiểu int
p =&v;     // gán địa chỉ của v cho con trỏ p
v=3;        // gán giá trị 3 cho v
*p=7;      // gán giá trị 7 cho v
p=&w;    // gán địa chỉ của w cho con trỏ p
*p=12;  // gán giá trị 12 cho w
```

DEMO Con trỏ

Con trỏ hằng

Khai báo hằng:

```
const int result = 5;  // result là hằng
result = 10;    // sau đó gán lại giá trị thì C++ sẽ báo lỗi
```

Khai báo con trỏ hằng:

```
const char* answer_ptr = "Forty-Two";
// answer_ptr là con trỏ trỏ tới hằng kiểu char
```

Dữ liệu được trỏ bởi con trỏ hằng thì không thể thay đổi nhưng con trỏ thì có thể.

```
answer_ptr = "Fifty-One"; // đúng (answer_ptr là biến con trỏ)
*answer_ptr = 'X'; // sai (*answer_ptr là hằng)
```

Con trỏ hằng

- Nếu khai báo:

```
char *const nameptr = "Test";    //name_ptr là con trỏ hằng
nameptr = "New"; // sai (name_ptr là hằng)
*nameptr = 'B'; // đúng (*nameptr là char)
```

- Nếu khai báo như sau thì không thể thay đổi được cả con trỏ và nội dung của con trỏ:

```
const char* const titleptr = "Title";
```


DEMO
DEMO Con trỏ hằng

Con trỏ và mảng

Mảng có thể được truy nhập thông qua con trỏ.

Tên mảng là một con trỏ hằng trỏ tới kiểu dữ liệu của các thành phần được lưu trữ trong mảng.

```
    int array[5] = { 23, 5, 12, 34, 17 };
// sử dụng chỉ số để truy nhập tới các phần tử của mảng
    for (int i=0; i< 5; i++)
        cout << array[i] << endl;
// sử dụng con trỏ để truy nhập tới các phần tử của mảng
    for (int i=0; i< 5; i++)
        cout << *(array+i) << endl;
```

Con trỏ và mảng

Cấp phát bộ nhớ cho con trỏ ta dùng toán tử new.

Ví dụ:

```
float* q;
q = new float;          // cấp phát bộ nhớ cho 1 số phảy động
*q = 3.14159; // *q đã được cấp phát
```

Ví dụ: `float* q = new float(3.14159);`

Trả về vùng nhớ đã được cấp phát để giải phóng bộ nhớ ta dùng toán tử delete.

Ví dụ:

```
float* q = new float(3.14159);
delete q;       // hủy cấp phát q
```

Ví dụ Con trỏ và mảng

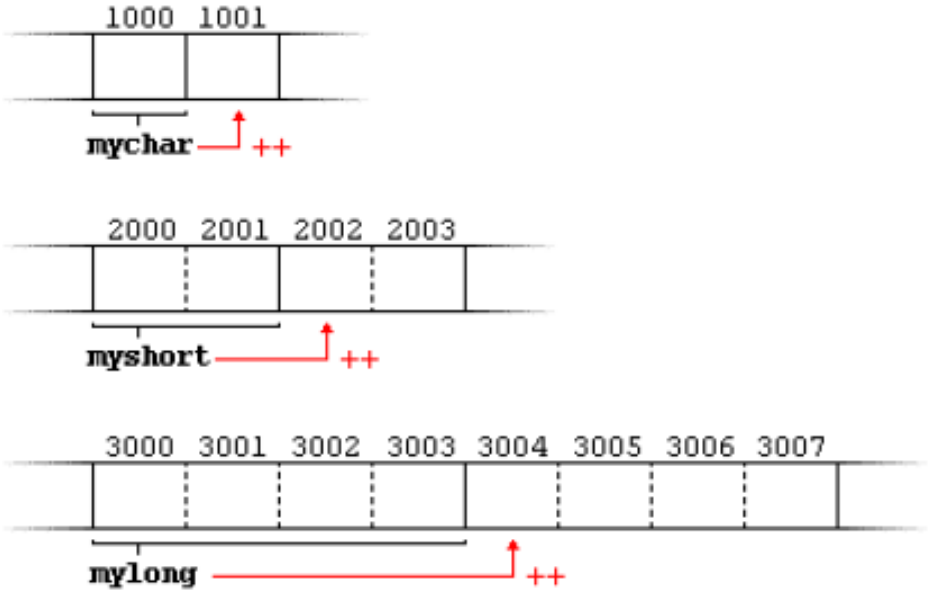
```
#include <iostream>
using namespace std;
int main ()
{
  int numbers[5];
  int * p;
  p = numbers; *p = 10;
  p++; *p = 20;
  p = &numbers[2]; *p = 30;
  p = numbers + 3; *p = 40;
  p = numbers; *(p+4) = 50;
  for (int n=0; n<5; n++)
        cout << numbers[n] << ", ";
  return 0;
}
```

DEMO Con trỏ và mảng

Phép toán học trên con trỏ

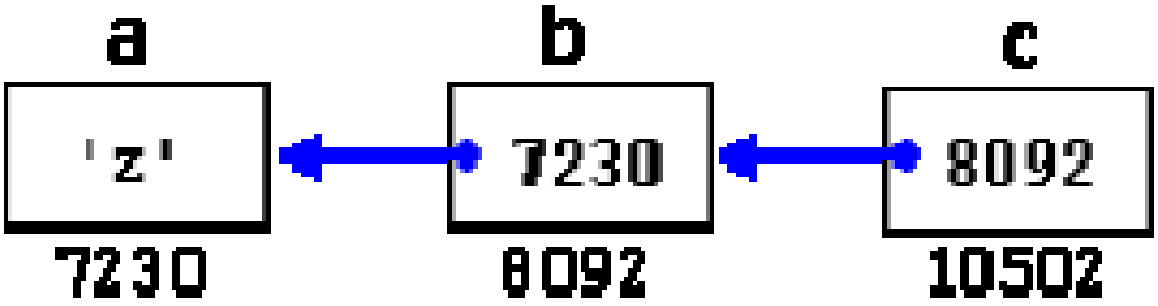
- Phân biệt các phép toán trên biến con trỏ và biến thông thường.

```
char *mychar;
short *myshort;
long *mylong;
mychar++;
myshort++;
mylong++;
```



Con trỏ trỏ tới con trỏ

```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```




DEMO
DEMO Con
trỏ tới con trỏ

Con trỏ void

Con trỏ void có thể lưu địa chỉ của biến có kiểu dữ liệu bất kỳ.

```
#include <iostream>
using namespace std;
void increase (void* data, int psize){
        if ( psize == sizeof(char) )
        { char* pchar; pchar=(char*)data; ++(*pchar); }
        else if (psize == sizeof(int) )
                { int* pint; pint=(int*)data; ++(*pint); }
}
int main (){
        char a = 'x';
        int b = 1602;
        increase (&a,sizeof(a));
        increase (&b,sizeof(b));
        cout << a << ", " << b << endl;
        return 0;
}
```

Con trỏ null

Con trỏ có giá trị null tức là không trỏ vào một địa chỉ nhớ cụ thể nào.

Con trỏ hàm

- Thường được sử dụng để truyền hàm như một tham số của hàm khác.
- Khi khai báo tham số là con trỏ hàm cần khai báo:
  - Prototype của hàm
  - Không sử dụng tên hàm
  - Đặt dấu * trước tên tham số

Con trỏ hàm

Ví dụ

```
#include <iostream>
using namespace std;

int addition (int a, int b)
{ return (a+b); }

int subtraction (int a, int b)
{ return (a-b); }

int operation (int x, int y, int *functocall)(int,int))
{
        int g;
        g = (*functocall)(x,y);
        return (g);
}
```

```
int main ()
{
        int m,n;
        int (*minus)(int,int) = subtraction;
        m = operation (7, 5, addition);
        n = operation (20, m, minus);
        cout <<n;
        return 0;
}
```

DEMO Con trỏ
void, hàm,null

Bộ Nhớ Động - Tĩnh

- Bộ nhớ động (heap)
  - Vùng nhớ được cấp phát động trong thời gian thực thi
- Bộ nhớ tính (stack)
  - Vùng nhớ được sử dụng để lưu trữ các biến toàn cục và lời gọi hàm
- Hai toán tử được sử dụng
  - **new**: cấp phát
  - **delete**: thu hồi

```
void Foo (void)
{
    int     *ptr = new int;
    char  *str = new char[10];
    //...
   delete  ptr;
   delete  [ ]str;
}
```

Tham Chiếu

- Một tham chiếu (reference) là một biệt hiệu (alias) cho một đối tượng.
- Cú pháp: *type*&**&** *ref-name* **=** *var-name*;

- Ví dụ

```
double num1 = 3.14;
double &num2 = num1;
```

num1 | 3.14 | *1000*

num2

- Ghi chú
  - Một tham chiếu phải luôn được khởi tạo khi nó được định nghĩa
  - Có thể khởi tạo tham chiếu tới một hằng

Truyền Bằng Trị - Con Trỏ - Tham Chiếu

```
   // Truyền bằng trị (đối tượng)
1  void Swap1 (int x, int y)
2  {
3     int temp = x;
4     x = y;
5     y = temp;
6  }
   // Truyền bằng địa chỉ (con trỏ)
7  void Swap2 (int *x, int *y)
8  {
9     int temp = *x;
10    *x = *y;
11    *y = temp;
   }
12 // Truyền bằng tham chiếu
13 void Swap3 (int &x, int &y)
14 {
15    int temp = x;
16    x = y;
17    y = temp;
18 }
```

```
int main (void)
{
   int i = 10, j = 20;
   Swap1(i, j);     cout << i << ", " << j << '\n';
   Swap2(&i, &j); cout << i << ", " << j << '\n';
   Swap3(i, j);     cout << i << ", " << j << '\n';
}
```

Giá trị của i,j tương ứng trong mỗi Swap1,2,3?

DEMO tham chiếu

Mảng động

Tạo mảng ta dùng toán tử new

- Cấp phát động cho biến con trỏ.
- Sau đó dùng nó như mảng chuẩn một chiều.

Ví dụ:

```
typedef double * DoublePtr;
DoublePtr d;
d = new double[10]; // kích thước trong cặp ngoặc vuông
```

Xóa mảng ta dùng câu lệnh delete[]

- Giải phóng tất cả vùng nhớ của mảng động
- Cặp ngoặc vuông báo hiệu có mảng

Ví dụ: sau khi cấp phát theo ví dụ tạo mảng trên ta dùng lệnh

```
delete [] d; d=NULL;
```

Xóa mảng. Chú ý sau khi xóa cần gán d=NULL;

Biến mảng động

- Nhắc lại: mảng lưu trong các ô nhớ liên tiếp
  - Biến mảng tham chiếu tới phần tử đầu tiên
  - Biến mảng là một kiểu biến con trỏ
- Ví dụ:

  int a[10];

  int *p;

  a,p đều là biến con trỏ.

DEMO mảng động

Kết luận

- Con trỏ và tham chiếu
- Con trỏ hằng
- Con trỏ hàm
- Mảng và con trỏ

Chuẩn bị bài sau

Sinh viên đọc sách và slide trước bài học kế tiếp về Lớp gồm:

- Tổng quan về Lớp
- Hàm thành viên nội tuyến (inline)
- Hàm xây dựng, hàm hủy, hàm bạn, lớp bạn
- Đối số mặc định,toán tử phạm vi
- Danh sách khởi tạo thành viên
- Thành viên hằng, tĩnh tham chiếu
- Thành viên là đối tượng của 1 lớp
- Mảng các đối tượng
- Cấu trúc (structure) và hợp (union)

Fpt University

FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn