

Bài 2.1: Con trỏ, địa chỉ và cấp phát động

- ✓ Địa chỉ của biến
- ✓ Con trỏ
- ✓ Con trỏ struct/class
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Địa chỉ của biến

- Tên biến là tên vùng nhớ dùng để lưu trữ dữ liệu tạm thời phục vụ hoạt động của chương trình.
- Vùng nhớ được cấp phát là vùng nhớ trong bộ nhớ RAM của thiết bị.
- Độ lớn của vùng nhớ tùy thuộc vào kiểu của biến. Ví dụ kiểu int là 4byte, double 8byte, char 1byte...
- Vùng nhớ kích thước càng lớn càng chứa được giá trị lớn. Ví dụ kiểu int lưu trữ giá trị tối đa khoảng 2.1 tỉ. Kiểu double lưu được giá trị lên tới $1.7E+308$.
- Vùng nhớ trong RAM là có hạn, có địa chỉ. Do đó địa chỉ của vùng nhớ cũng chính là địa chỉ của biến đang gắn với vùng nhớ đó.
- Để lấy địa chỉ của biến, ta thêm toán tử địa chỉ & trước tên biến.
- Địa chỉ của biến thường biểu diễn ở hệ 16 hoặc hệ 10 tùy hệ điều hành.

Ví dụ

➤ Sau đây là ví dụ lấy địa chỉ của biến trong C++:

```
int number1 = 300;
int number2 = 260;
int sum = number1 + number2;
// hiển thị địa chỉ và giá trị chứa trong địa chỉ
// của vùng nhớ tương ứng:
cout << left << setw(15) << "Địa chỉ"
    << setw(12) << "Giá trị" << endl;
cout << left << setw(15) << &number1
    << setw(12) << number1 << endl;
cout << left << setw(15) << &number2
    << setw(12) << number2 << endl;
cout << left << setw(15) << &sum
    << setw(12) << sum << endl;
```

Dia chi	Gia tri
008FF828	300
008FF81C	260
008FF810	560

Con trỏ

- Con trỏ là biến mà giá trị chứa trong bản thân nó là địa chỉ của một biến khác.
- Con trỏ được sử dụng rộng rãi trong lập trình C/C++ vì tính linh hoạt, mềm dẻo của nó.
- Ứng dụng của con trỏ: tạo mảng cấp phát động, tạo danh sách liên kết, cây nhị phân...
- Vì biết được địa chỉ của biến nên con trỏ có thể thao tác trực tiếp để thay đổi giá trị tại vùng nhớ nó đang trỏ tới.

Cú pháp khai báo con trỏ

Cú pháp với con trỏ: **type* pointerName;**

Trong đó:

- Type là kiểu dữ liệu hợp lệ trong C++ như void, int, Student...
- Dấu * là bắt buộc. Nó giúp ta phân biệt biến thường vs biến con trỏ. Có thể để * ngay sau tên kiểu hoặc trước tên con trỏ.
- pointerName là tên con trỏ. Đặt như quy tắc đặt tên biến và thường thêm hậu tố ptr để dễ nhận biết đó là biến con trỏ.
- Khuyến nghị chỉ khai báo 1 con trỏ trên 1 dòng.

```
int* aPtr;           // con trỏ kiểu int
string* strPtr;      // con trỏ kiểu string
double* xPtr;        // con trỏ kiểu double
char* namePtr;       // con trỏ kiểu char
```

Gán và khởi tạo con trỏ

- Con trỏ phải được khai báo và gán giá trị trước khi nó được sử dụng.
- Về khởi tạo: thường khởi tạo cho con trỏ giá trị null, cú pháp là NULL, 0 hoặc nullptr(từ C++ 11). Hoặc lấy địa chỉ của biến cụ thể cùng kiểu với kiểu của con trỏ.
- Giá trị gán cho con trỏ có thể là null, địa chỉ của biến cùng kiểu hoặc con trỏ cùng kiểu với con trỏ cần gán.
- Giá trị 0 là giá trị nguyên duy nhất được gán cho con trỏ. Thực tế ta tránh sử dụng giá trị này vì dễ nhầm lẫn với giá trị của biến thường.
- Ý nghĩa của giá trị null: khi con trỏ trỏ đến null tức là con trỏ không có giá trị xác định. Nó không có ý nghĩa khi thực hiện bất kì tham chiếu nào từ con trỏ.

Ví dụ

```
int a = 300;
int b = 260;
int sum = a + b;
string msg = "Hello pointer, I'll love you soon.";

int* aPtr = &a; // khởi tạo aPtr trỏ đến biến a
int* sumPtr = nullptr; // khởi tạo sumPtr trỏ đến null
long* otherPtr = nullptr; // ok, otherPtr trỏ đến null
string* msgPtr = &msg; // ok, msgPtr trỏ đến biến msg

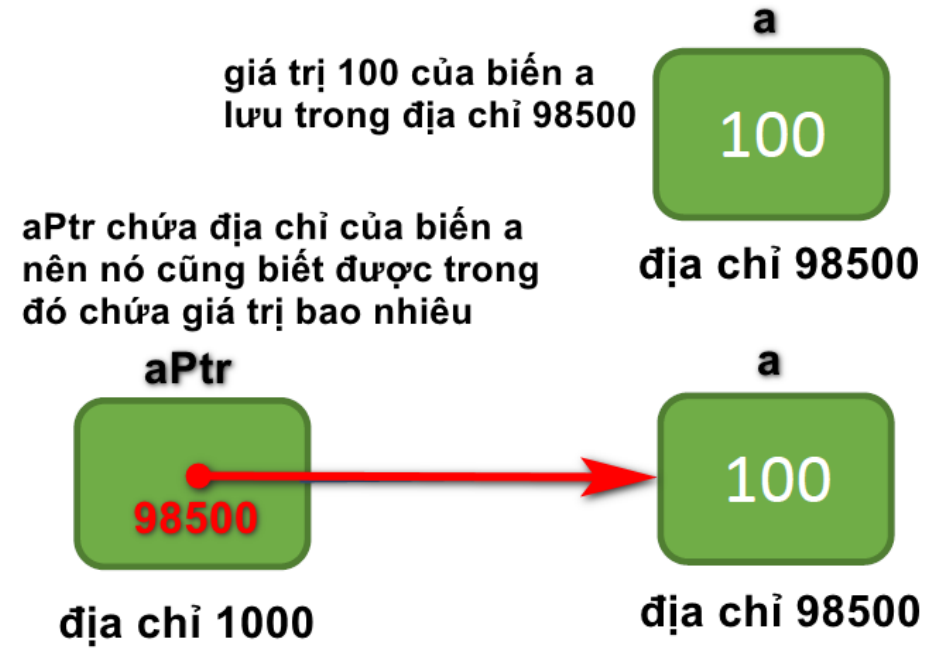
// kiểm tra xem có đúng là con trỏ trỏ đến
// địa chỉ mà nó được gán hay không:
cout << "Địa chỉ biến a: " << &a << endl;
cout << "Giá trị trong aPtr: " << aPtr << endl;
cout << "Địa chỉ biến msg: " << &msg << endl;
cout << "Giá trị trong msgPtr: " << msgPtr << endl;
// kiểm tra xem giá trị của con trỏ null có đúng là 0?
cout << "Giá trị của con trỏ null: " << nullptr << endl;
```

Các toán tử với con trỏ

Có 2 toán tử: `&`, `*`

- Toán tử `&` gọi là toán tử địa chỉ. Dùng để lấy địa chỉ của một biến.
- Khi sử dụng `&` với tên con trỏ ta sẽ có được địa chỉ của con trỏ.
- Toán tử `&` chỉ sử dụng với tên biến, tên con trỏ. Không sử dụng với các giá trị cụ thể vì sẽ bị lỗi.
- Khi ta gán địa chỉ một biến hoặc giá trị null cho con trỏ. Ta nói rằng con trỏ trỏ tới biến đó hoặc trỏ tới null tương ứng.
- Toán tử `*` đặt trước tên con trỏ, gọi là toán tử phân giải, toán tử gián tiếp. Dùng để lấy giá trị trực tiếp từ địa chỉ mà con trỏ đang trỏ tới.
- Tránh nhầm lẫn toán tử gián tiếp và cú pháp khai báo con trỏ. Trong cú pháp khai báo con trỏ có sự xuất hiện của kiểu. Trong cú pháp phân giải địa chỉ của con trỏ không có tên kiểu dữ liệu.
- Khi sử dụng toán tử `*` cần đảm bảo rằng con trỏ ta đang thao tác là khác null.

Minh họa



Minh họa

```
int a = 300;
int b = 260;
int sum = a + b;
string msg = "Hello pointer, I'll love you soon.";

int* aPtr = &a; // khởi tạo aPtr trỏ đến biến a
int* sumPtr = nullptr; // khởi tạo sumPtr trỏ đến null
long* otherPtr = nullptr; // ok, otherPtr trỏ đến null
string* msgPtr = &msg; // ok, msgPtr trỏ đến biến msg

// ví dụ sử dụng toán tử &, *
cout << "Địa chỉ biến a: " << &a
      << ", giá trị: " << a << endl;
cout << " Địa chỉ lưu trong aPtr: " << aPtr
      << ", giá trị tại địa chỉ này: " << *aPtr << endl;
cout << "Địa chỉ biến msg: " << &msg
      << ", giá trị: " << msg << endl;
cout << "Địa chỉ lưu trong msgPtr: " << msgPtr
      << ", giá trị tại địa chỉ này: " << *msgPtr << endl;
```

Con trỏ struct/class

- Khi ta tạo struct hoặc class ta cũng có thể sử dụng biến con trỏ để sử dụng.
- Ta có thể cấp phát động cho con trỏ này trong C++ qua cú pháp:
`pointerName = new structName();`
- Với ngôn ngữ C ta sử dụng malloc:
`pointerName = (structName*)malloc(sizeof(structName));`
- Sau đó, để truy cập đến một thành phần trong đối tượng của struct/class, ta sử dụng kí hiệu `->` thay vì dấu chấm.

Minh họa trong C++

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

typedef struct {
    string id; // mã sinh viên
    string fullName; // họ tên
    int age; // tuổi
} Student;

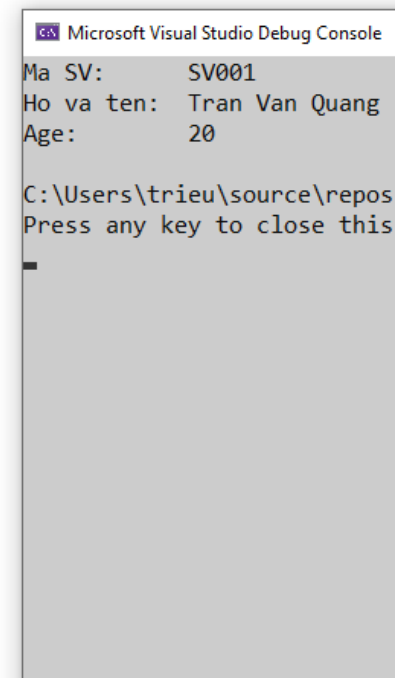
int main()
{
    Student* sPtr = new Student();
    // gán thông tin:
    sPtr->id = "SV001";
    sPtr->fullName = "Tran Van Quang";
    sPtr->age = 20;
    // lấy giá trị ra
    cout << "Ma SV: " << sPtr->id << endl;
    cout << "Ho va ten: " << sPtr->fullName << endl;
    cout << "Age: " << sPtr->age << endl;
    return 0;
}
```

Minh họa trong C

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

typedef struct {
    char id[20]; // mã sinh viên
    char fullName[40]; // họ tên
    int age; // tuổi
} Student;

int main()
{
    Student* sPtr = (Student*) malloc(sizeof(Student));
    // gán thông tin:
    strcpy(sPtr->id, "SV001");
    strcpy(sPtr->fullName, "Tran Van Quang");
    sPtr->age = 20;
    // lấy giá trị ra
    printf("%-12s%s\n", "Ma SV: ", sPtr->id);
    printf("%-12s%s\n", "Ho va ten: ", sPtr->fullName);
    printf("%-12s%d\n", "Age: ", sPtr->age);
    return 0;
}
```



Microsoft Visual Studio Debug Console

```
Ma SV:      SV001
Ho va ten:  Tran Van Quang
Age:       20

C:\Users\trieu\source\repos
Press any key to close this
```

Nội dung tiếp theo

Mảng một chiều