



Phần trình bày của:

ĐẬU HẢI PHONG

Giảng viên

Đại Nam, ngày 19 tháng 01 năm 2023

LƯU Ý

**KHÔNG NÓI
CHUYỆN RIÊNG**



**KHÔNG SỬ DỤNG
ĐIỆN THOẠI**



KHÔNG NGỦ GẬT



GHI CHÉP ĐẦY ĐỦ



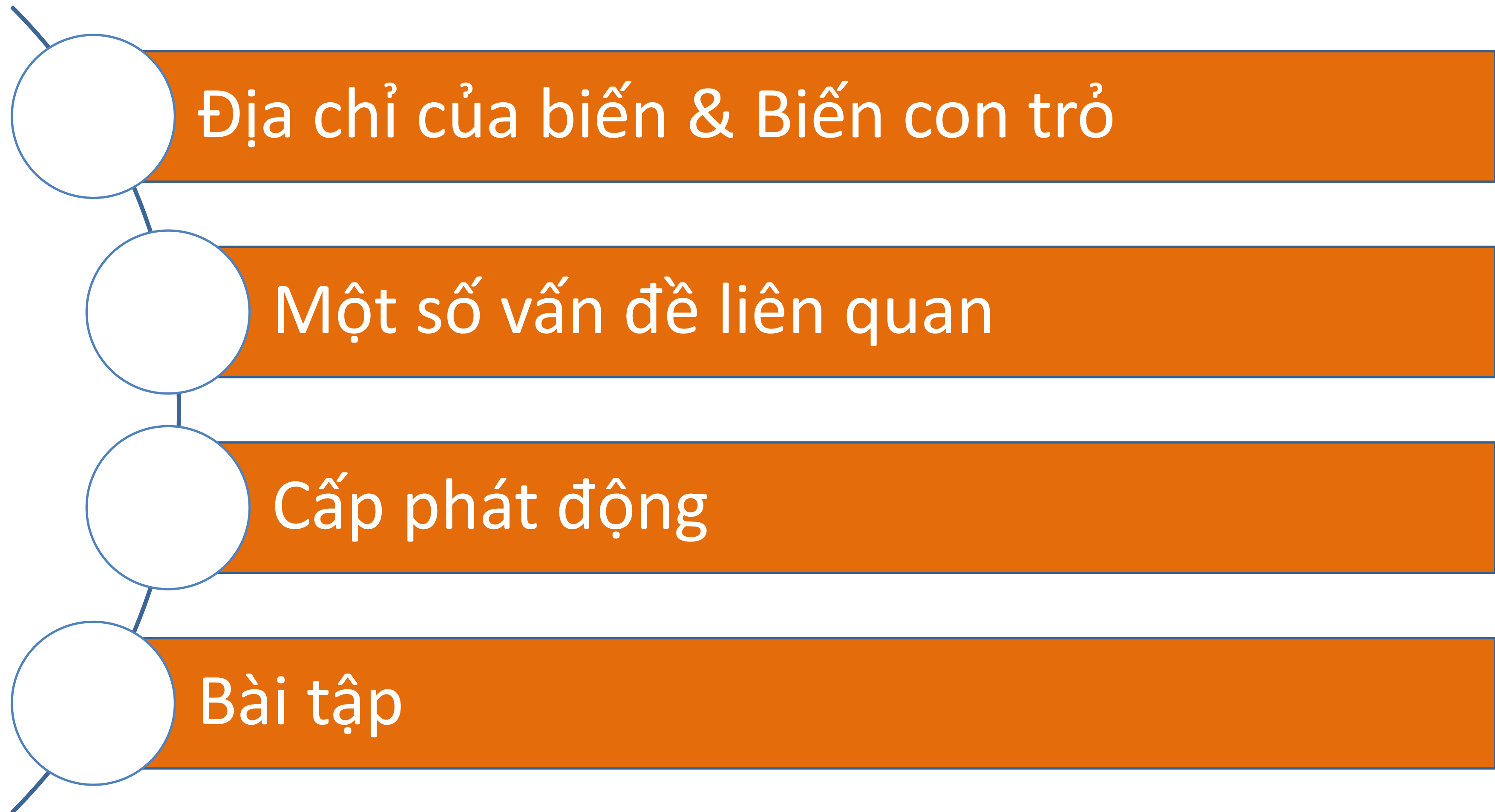


LẬP TRÌNH CƠ BẢN



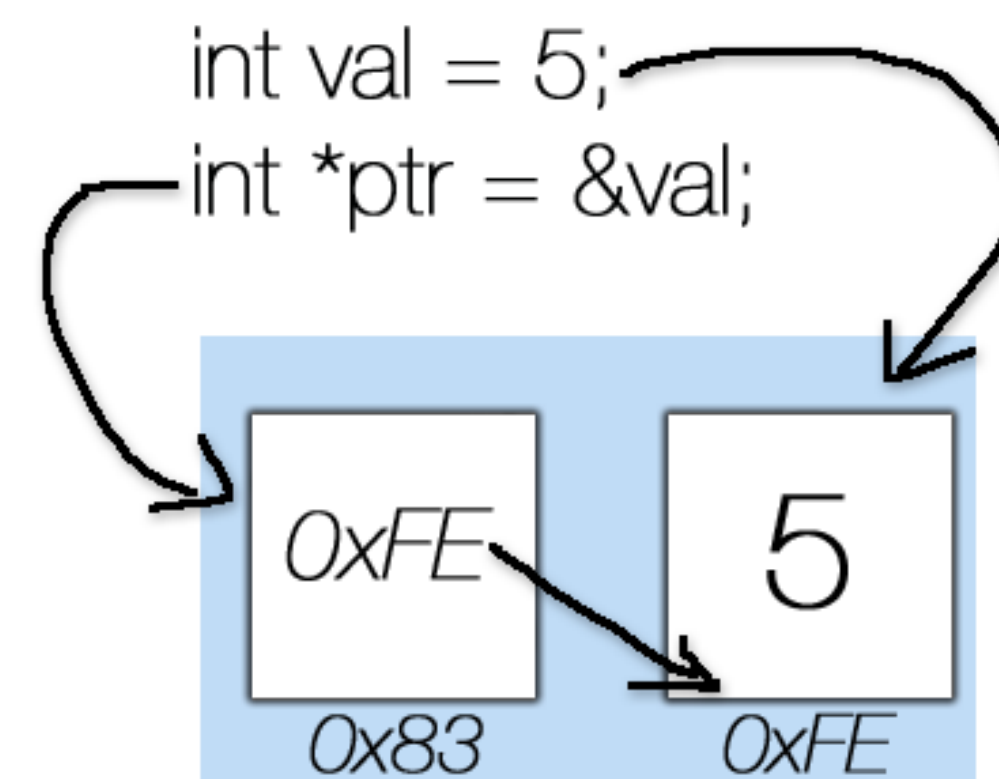
CHU'ƠNG 8

CON TRỎ



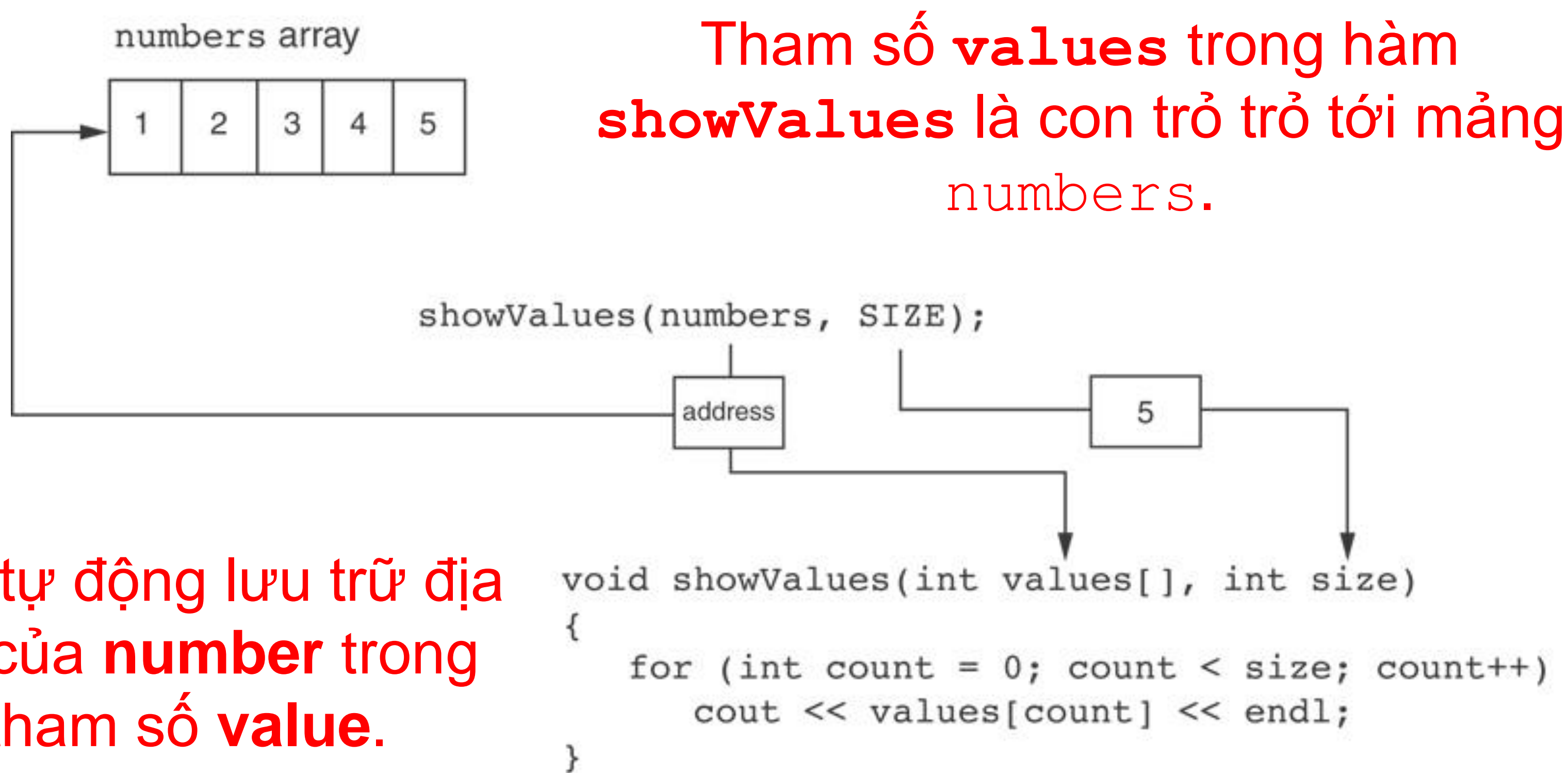
Địa chỉ biến & Biến con trỏ

- Địa chỉ biến:
 - Mỗi biến trong chương trình được lưu trong một địa chỉ duy nhất.
 - Sử dụng toán tử địa chỉ & để lấy địa chỉ của biến
 - `int num = -99;`
 - `cout<<# //In ra địa chỉ của biến num hệ hecxa`
- Biến con trỏ:
 - Thường được gọi là con trỏ, là 1 biến dùng để *nắm giữ địa chỉ*
 - Bởi vì biến con trỏ nắm giữ địa chỉ của phần dữ liệu khác hay nó “trỏ” đến dữ liệu



Một số thứ giống như con trỏ

- Truyền mảng như đối số của hàm
 - `showValues(numbers, SIZE);`



Một số thứ giống như con trỏ

- Biến tham chiếu:

- Giả sử có hàm sau:

```
void getOrder(int &donuts)
{
    cout << "How many doughnuts do you want? ";
    cin >> donuts;
}
```

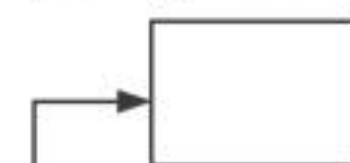
- Và gọi hàm như sau:

```
int jellyDonuts;
getOrder(jellyDonuts);
```

C++ tự động lưu trữ địa chỉ của
jellyDonuts trong tham số donuts.

Tham số donuts trong hàm
getOrder trở tới biến jellyDonuts.

jellyDonuts variable



getOrder(jellyDonuts);

address

void getOrder(int &donuts)

```
{
    cout << "How many doughnuts do you want? ";
    cin >> donuts;
}
```


Biến con trỏ

- Biến con trỏ là một cách dùng 1 *địa chỉ bộ nhớ để làm việc* với một phần dữ liệu.
- Con trỏ là mức thấp hơn (low-level) mẫn và tham chiếu
- Điều này có nghĩa là bạn *phải chịu trách nhiệm tìm địa chỉ* bạn muốn lưu trong con trỏ và sử dụng nó chính xác.
- Định nghĩa:
 `int *intptr;`
 → Hiểu là intptr có thể nắm giữ địa chỉ của biến int
- Khoảng cách trong định nghĩa không quan trọng:
 `int * intptr; // giống ở trên`
 `int* intptr; // giống ở trên`

Biến con trỏ

- Gán 1 địa chỉ cho biến con trỏ & Tổ chức bộ nhớ:

```
int *intptr;
```

```
intptr = &num;
```

Program 9-2

```
1 // This program stores the address of a variable in a pointer.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int x = 25;    // int variable
8     int *ptr;      // Pointer variable, can point to an int
9
10    ptr = &x;      // Store the address of x in ptr
11    cout << "The value in x is " << x << endl;
12    cout << "The address of x is " << ptr << endl;
13    return 0;
14 }
```

num

25

intptr

0x4a00

Địa chỉ của num: 0x4a00

Program Output

```
The value in x is 25
The address of x is 0x7e00
```

Thao tác gián tiếp

- Toán tử gián tiếp (*) tham chiếu đến một con trỏ
- Cho phép truy cập vào phần tử mà con trỏ trỏ đến

```
int x = 25;  
int *intptr = &x;  
cout << *intptr << endl; //In ra 25
```

Program Output

```
Here is the value in x, printed twice:  
25  
25  
Once again, here is the value in x:  
100  
100
```

Program 9-3

```
1  // This program demonstrates the use of the indirection operator.  
2  #include <iostream>  
3  using namespace std;  
4  
5  int main()  
6  {  
7      int x = 25;      // int variable  
8      int *ptr;        // Pointer variable, can point to an int  
9  
10     ptr = &x;         // Store the address of x in ptr  
11  
12     // Use both x and ptr to display the value in x.  
13     cout << "Here is the value in x, printed twice:\n";  
14     cout << x << endl;    // Displays the contents of x  
15     cout << *ptr << endl; // Displays the contents of x  
16  
17     // Assign 100 to the location pointed to by ptr. This  
18     // will actually assign 100 to x.  
19     *ptr = 100;  
20  
21     // Use both x and ptr to display the value in x.  
22     cout << "Once again, here is the value in x:\n";  
23     cout << x << endl;    // Displays the contents of x  
24     cout << *ptr << endl; // Displays the contents of x  
25     return 0;  
26 }
```

Mối quan hệ giữa mảng và con trỏ

- Tên mảng bắt đầu bằng địa chỉ của mảng

```
int vals[] = {4, 7, 11};
```

| | | |
|---|---|----|
| 4 | 7 | 11 |
|---|---|----|

- Địa chỉ của vals: 0xa00

```
cout << vals;           // displays 0xa00
```

```
cout << vals[0];        // displays 4
```

- Tên mảng có thể sử dụng như con trỏ hằng:

```
int vals[] = {4, 7, 11};
```

```
cout << *vals;          // displays 4
```

- Con trỏ có thể sử dụng như tên mảng:

```
int *valptr = vals;
```

```
cout << valptr[1];       // displays 7
```

Mối quan hệ giữa mảng và con trỏ

Program 9-5

```
1  // This program shows an array name being dereferenced with the *
2  // operator.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8      short numbers[] = {10, 20, 30, 40, 50};
9
10     cout << "The first element of the array is ";
11     cout << *numbers << endl;
12     return 0;
13 }
```

Program Output

The first element of the array is 10

Con trỏ trong biểu thức

- Giả sử:

```
int vals[]={4,7,11}, *valptr;  
valptr = vals;
```
- Vậy `valptr + 1` là gì???
 - Điều này có nghĩa là (địa chỉ `valptr`) + (1*kích thước `int`);

```
cout << *(valptr+1); //displays 7  
cout << *(valptr+2); //displays 11
```
- Chú ý là sử dụng `()` trong biểu thức

Truy cập mảng

- Các phần tử của mảng có thể truy cập bằng một số cách sau:

| Phương pháp truy cập mảng | Ví dụ |
|---|----------------------------------|
| Tên mảng và [] | <code>vals[2] = 17;</code> |
| Con trỏ trỏ đến mảng và [] | <code>valptr[2] = 17;</code> |
| Tên mảng và chỉ số của mảng | <code>*(vals + 2) = 17;</code> |
| Con trỏ trỏ đến mảng và chỉ số của mảng | <code>*(valptr + 2) = 17;</code> |

- Chuyển đổi: `vals[i]` tương đương với `*(vals + i)`
- Khi sử dụng tên mảng hay con trỏ thì: Không thực hiện kiểm tra giới hạn mảng trong truy cập mảng

Truy cập mảng

Program 9-7

```
1 // This program uses subscript notation with a pointer variable and
2 // pointer notation with an array name.
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     const int NUM_COINS = 5;
9     double coins[NUM_COINS] = {0.05, 0.1, 0.25, 0.5, 1.0};
10    double *doublePtr;    // Pointer to a double
11    int count;            // Array index
12
13    // Assign the address of the coins array to doublePtr.
14    doublePtr = coins;
15
16    // Display the contents of the coins array. Use subscripts
17    // with the pointer!
18    cout << "Here are the values in the coins array:\n";
19    for (count = 0; count < NUM_COINS; count++)
20        cout << doublePtr[count] << " ";
21
22    // Display the contents of the array again, but this time
23    // use pointer notation with the array name!
24    cout << "\nAnd here they are again:\n";
25    for (count = 0; count < NUM_COINS; count++)
26        cout << *(coins + count) << " ";
27    cout << endl;
28    return 0;
29 }
```

Program Output

```
Here are the values in the coins array:
0.05 0.1 0.25 0.5 1
And here they are again:
0.05 0.1 0.25 0.5 1
```

Phép toán trên con trỏ

- Thao tác trên biến con trỏ

| Thao tác | Ví dụ |
|--------------------------|--|
| | <pre>int vals[]={4,7,11}; int *valptr = vals;</pre> |
| ++, -- | <pre>valptr++; // points at 7 valptr--; // now points at 4</pre> |
| +, - (pointer and int) | <pre>cout << *(valptr + 2); // 11</pre> |
| +=, -= (pointer and int) | <pre>valptr = vals; // points at 4 valptr += 2; // points at 11</pre> |
| - (pointer from pointer) | <pre>cout << valptr-val; // difference // (number of ints) between valptr // and val</pre> |

Phép toán trên con trỏ

Program 9-9

```
1 // This program uses a pointer to display the contents of an array.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     const int SIZE = 8;
8     int set[SIZE] = {5, 10, 15, 20, 25, 30, 35, 40};
9     int *numPtr = nullptr; // Pointer
10    int count;             // Counter variable for loops
11
12    // Make numPtr point to the set array.
13    numPtr = set;
14
15    // Use the pointer to display the array contents.
16    cout << "The numbers in set are:\n";
17    for (count = 0; count < SIZE; count++)
18    {
19        cout << *numPtr << " ";
20        numPtr++;
21    }
22
23    // Display the array contents in reverse order.
24    cout << "\nThe numbers in set backward are:\n";
25    for (count = 0; count < SIZE; count++)
26    {
27        numPtr--;
28        cout << *numPtr << " ";
29    }
30    return 0;
31 }
```

Program Output

```
The numbers in set are:
5 10 15 20 25 30 35 40
The numbers in set backward are:
40 35 30 25 20 15 10 5
```


Kiểm tra



CHECKPOINT

Checkpoint

9.1 Write a statement that displays the address of the variable `count`.

9.2 Write the definition statement for a variable `fltPtr`. The variable should be a pointer to a `float`.

9.3 List three uses of the `*` symbol in C++.

9.4 What is the output of the following code?

```
int x = 50, y = 60, z = 70;
int *ptr = nullptr;

cout << x << " " << y << " " << z << endl;
ptr = &x;
*ptr *= 10;
ptr = &y;
*ptr *= 5;
ptr = &z;
*ptr *= 2;
cout << x << " " << y << " " << z << endl;
```

9.5 Rewrite the following loop so it uses pointer notation (with the indirection operator) instead of subscript notation.

```
for (int x = 0; x < 100; x++)
    cout << arr[x] << endl;
```

9.6 Assume `ptr` is a pointer to an `int` and holds the address 12000. On a system with 4-byte integers, what address will be in `ptr` after the following statement?

```
ptr += 10;
```

9.7 Assume `ptr` is a pointer variable. Is each of the following statements valid or invalid? If any is invalid, why?

- A) `ptr++;`
- B) `--ptr;`
- C) `ptr /= 2;`
- D) `ptr *= 4;`
- E) `ptr += x; // Assume x is an int.`

9.8 Is each of the following definitions valid or invalid? If any is invalid, why?

- A) `int ivar;`
`int *iptr = &ivar;`
- B) `int ivar, *iptr = &ivar;`
- C) `float fvar;`
`int *iptr = &fvar;`
- D) `int nums[50], *iptr = nums;`
- E) `int *iptr = &ivar;`
`int ivar;`

Khởi tạo và So sánh con trỏ

- Khởi tạo:

- Có thể khởi tạo khi định nghĩa

```
int num, *numptr = &num;
```

```
int val[3], *valptr = val;
```

- Không thể sai kiểu dữ liệu

```
double cost;
```

```
int *ptr = &cost; // sẽ không làm việc SAI
```

- Có thể kiểm tra con trỏ **ptr** không hợp lệ với cú pháp sau:

```
if (!ptr) ...
```

- So sánh:

- Toán tử quan hệ: >, <, =, ... có thể sử dụng để so sánh địa chỉ con trỏ
- So sánh địa chỉ của con trỏ không giống như nội dung được trỏ bởi con trỏ.

- `if (ptr1 == ptr2) // so sánh địa chỉ`
- `if (*ptr1 == *ptr2) // so sánh nội dung`

Con trỏ là tham số của hàm

- Một con trỏ có thể là 1 tham số
- Làm việc *giống như biến tham chiếu*, cho phép thay đổi đối số từ trong hàm.
- Yêu cầu:
 - Có dấu * trong hàm nguyên mẫu và tiêu đề hàm
`void getNum(int *ptr);` // ptr là con trỏ trỏ đến kiểu nguyên
 - Có dấu * trong thân hàm để tham chiếu đến con trỏ
`cin >> *ptr;`
 - Truyền địa chỉ của đối số cho hàm
`getNum(&num);` // *truyền địa chỉ của num* cho hàm `getNum`

• Ví dụ:

```
void swap(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

```
int num1 = 2, num2 = -3;
swap(&num1, &num2);
```


Con trỏ là tham số của hàm

Program 9-11

```
1 // This program uses two functions that accept addresses of
2 // variables as arguments.
3 #include <iostream>
4 using namespace std;
5
6 // Function prototypes
7 void getNumber(int *);
8 void doubleValue(int *);
9
10 int main()
11 {
12     int number;
13
14     // Call getNumber and pass the address of number.
15     getNumber(&number);
16
17     // Call doubleValue and pass the address of number.
18     doubleValue(&number);
19
20     // Display the value in number.
21     cout << "That value doubled is " << number << endl;
22     return 0;
23 }
24
```

```
25 //*****
26 // Definition of getNumber. The parameter, input, is a pointer. *
27 // This function asks the user for a number. The value entered *
28 // is stored in the variable pointed to by input.
29 //*****
30
31 void getNumber(int *input)
32 {
33     cout << "Enter an integer number: ";
34     cin >> *input;
35 }
36
37 //*****
38 // Definition of doubleValue. The parameter, val, is a pointer. *
39 // This function multiplies the variable pointed to by val by *
40 // two.
41 //*****
42
43 void doubleValue(int *val)
44 {
45     *val *= 2;
46 }
```

Program Output with Example Input Shown in Bold

Enter an integer number: **10**
That value doubled is 20

Con trỏ hằng

- Nếu muốn lưu địa chỉ của một hằng trong một con trỏ, thì cần lưu nó trong một con trỏ trỏ đến hằng.
- Ví dụ:

```
const int SIZE = 6;
```

```
const double payRates[SIZE]={18.55, 17.45, 12.85, 14.97, 10.35, 18.89};
```

- Trong đó, **payRates** là mảng của hằng số double.
- Truyền mảng **payRates** cho hàm:

```
void displayPayRates(const double *rates, int size)
{
    for (int count = 0; count < size; count++)
    {
        cout << "Pay rate for employee " << (count + 1)
              << " is $" << *(rates + count) << endl;
    }
}
```

Tham số **rates** là con trỏ trỏ tới hằng **double**.

Con trỏ hằng

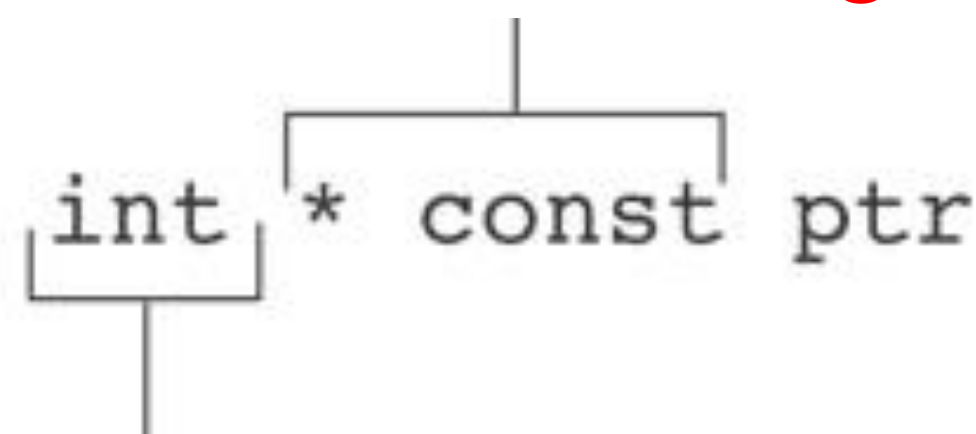
- Một con trỏ hằng là một con trỏ mà được khởi tạo với một địa chỉ thì **không thể** trỏ đến bất cứ đâu.

– Ví dụ:

```
int value = 22
```

```
int * const ptr
```

* **const** xác định ptr là một con trỏ hằng



Đây là xác định cái gì mà ptr trỏ đến

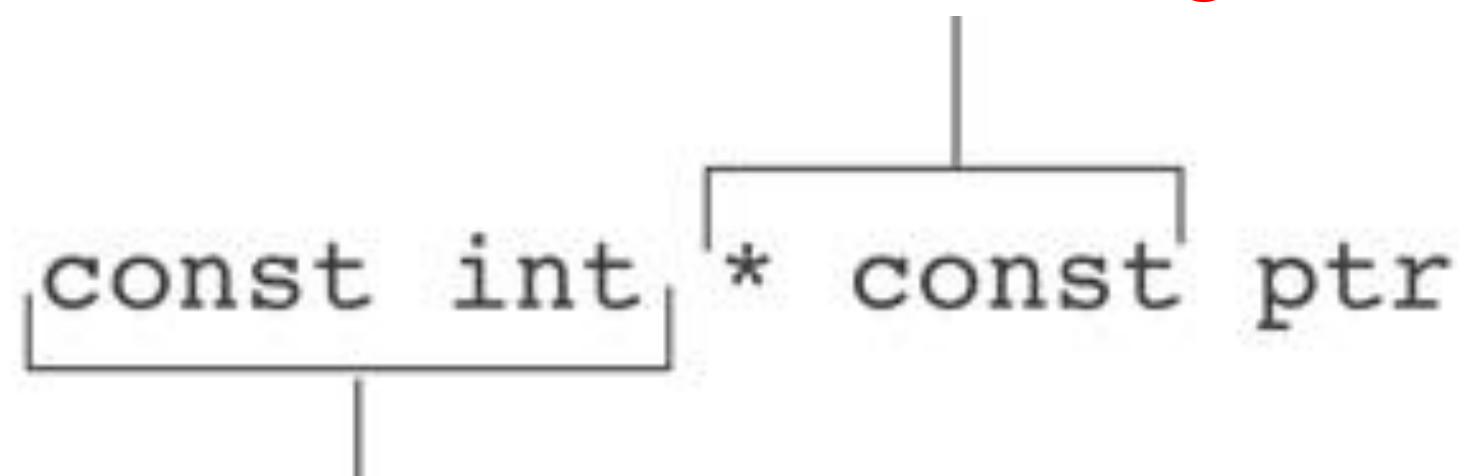
Con trỏ hằng trỏ đến hằng số

- Một con trỏ hằng trỏ đến hằng số:
 - Là 1 con trỏ mà con trỏ đó trỏ đến hằng số
 - Một con trỏ không thể trỏ đến bất kỳ đâu ngoại trừ những gì nó đã trỏ đến.
- Ví dụ:

```
int value = 22;
```

```
const int * const ptr = &value;
```

* **const** xác định ptr là
một con trỏ hằng



Đây là xác định cái gì
mà ptr trỏ đến

Cấp phát bộ nhớ động

- Có thể cấp phát lưu trữ cho 1 biến khi chương trình đang chạy.
- Máy tính sẽ trả về địa chỉ của biến cấp phát mới.
- Sử dụng toán tử **new** để cấp phát bộ nhớ
double *dptr;
dptr = new double;
- **new** trả về địa chỉ bộ nhớ
- Có thể sử dụng new để cấp phát mảng
const int SIZE = 25;
arrayPtr = new double[SIZE];
- Có thể truy cập mảng như sau:
- Chương trình sẽ dừng khi không đủ bộ nhớ để cấp phát

```
for(i = 0; i < SIZE; i++)  
    *arrayptr[i] = i * i;  
or  
for(i = 0; i < SIZE; i++)  
    *(arrayptr + i) = i * i;
```

Giải phóng bộ nhớ

- Sử dụng delete để giải phóng bộ nhớ
 - delete fptr;
- Sử dụng [] để giải phóng mảng động
 - delete [] arrayptr;
- Chỉ sử dụng delete với cấp phát bộ nhớ động

Program 9-14

```
1  // This program totals and averages the sales figures for any
2  // number of days. The figures are stored in a dynamically
3  // allocated array.
4  #include <iostream>
5  #include <iomanip>
6  using namespace std;
7
8  int main()
9  {
10     double *sales,      // To dynamically allocate an array
11           total = 0.0,  // Accumulator
12           average;      // To hold average sales
```

Giải phóng bộ nhớ

Program 9-14 (continued)

```
13     int numDays,          // To hold the number of days of sales
14         count;            // Counter variable
15
16     // Get the number of days of sales.
17     cout << "How many days of sales figures do you wish ";
18     cout << "to process? ";
19     cin >> numDays;
20
21     // Dynamically allocate an array large enough to hold
22     // that many days of sales amounts.
23     sales = new double[numDays];
24
25     // Get the sales figures for each day.
26     cout << "Enter the sales figures below.\n";
27     for (count = 0; count < numDays; count++)
28     {
29         cout << "Day " << (count + 1) << ": ";
30         cin >> sales[count];
31     }
32
33     // Calculate the total sales
34     for (count = 0; count < numDays; count++)
35     {
36         total += sales[count];
37     }
38
39     // Calculate the average sales per day
40     average = total / numDays;
41
42     // Display the results
43     cout << fixed << showpoint << setprecision(2);
44     cout << "\n\nTotal Sales: $" << total << endl;
45     cout << "Average Sales: $" << average << endl;
46
47     // Free dynamically allocated memory
48     delete [] sales;
49     sales = 0;          // Make sales point to null.
50
51     return 0;
52 }
```

Dòng 49, gán 0 cho con trỏ sales sau khi giải phóng. Điều này tránh truy cập đến vùng nhớ sau giải phóng và tránh sử dụng lại

Program Output with Example Input Shown in Bold

```
How many days of sales figures do you wish to process? 5 [Enter]
Enter the sales figures below.
Day 1: 898.63 [Enter]
Day 2: 652.32 [Enter]
Day 3: 741.85 [Enter]
Day 4: 852.96 [Enter]
Day 5: 921.37 [Enter]

Total Sales: $4067.13
Average Sales: $813.43
```


Hàm trả về con trỏ

- Con trỏ có thể là kiểu trả về của hàm:
 - `int * newNum();`
- Hàm không được trả về một con trỏ trỏ đến một biến cục bộ trong hàm.
- Hàm trả về con trỏ khi:
 - Dữ liệu được truyền vào cho hàm như 1 đối số
 - Để bộ nhớ cấp phát động

Program 9-15

```
1 // This program demonstrates a function that returns
2 // a pointer.
3 #include <iostream>
4 #include <cstdlib> // For rand and srand
5 #include <ctime>   // For the time function
6 using namespace std;
7
8 // Function prototype
9 int *getRandomNumbers(int);
10
```

(program continues)

Hàm trả về con trỏ

Program 9-15

(continued)

```
11 int main()
12 {
13     int *numbers = nullptr; // To point to the numbers
14
15     // Get an array of five random numbers.
16     numbers = getRandomNumbers(5);
17
18     // Display the numbers.
19     for (int count = 0; count < 5; count++)
20         cout << numbers[count] << endl;
21
22     // Free the memory.
23     delete [] numbers;
24     numbers = nullptr;
25     return 0;
26 }
27
```

Program Output

```
2712
9656
24493
12483
7633
```

```
28 //*****
29 // The getRandomNumbers function returns a pointer *
30 // to an array of random integers. The parameter *
31 // indicates the number of numbers requested. *
32 //*****
33
34 int *getRandomNumbers(int num)
35 {
36     int *arr = nullptr; // Array to hold the numbers
37
38     // Return a null pointer if num is zero or negative.
39     if (num <= 0)
40         return nullptr;
41
42     // Dynamically allocate the array.
43     arr = new int[num];
44
45     // Seed the random number generator by passing
46     // the return value of time(0) to srand.
47     srand( time(0) );
48
49     // Populate the array with random numbers.
50     for (int count = 0; count < num; count++)
51         arr[count] = rand();
52
53     // Return a pointer to the array.
54     return arr;
55 }
```


Kiểm tra



9.9 Assuming arr is an array of ints, will each of the following program segments display “True” or “False”?

- A)

```
if (arr < &arr[1])
    cout << "True";
else
    cout << "False";
```
- B)

```
if (&arr[4] < &arr[1])
    cout << "True";
else
    cout << "False";
```
- C)

```
if (arr != &arr[2])
    cout << "True";
else
    cout << "False";
```
- D)

```
if (arr != &arr[0])
    cout << "True";
else
    cout << "False";
```

9.10 Give an example of the proper way to call the following function:

```
void makeNegative(int *val)
{
    if (*val > 0)
        *val = -(*val);
}
```

9.11 Complete the following program skeleton. When finished, the program will ask the user for a length (in inches), then convert that value to centimeters, and display the result. You are to write the function convert. (Note: 1 inch = 2.54 cm. Do not modify function main.)

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

// Write your function prototype here.
```

```
int main()
{
    double measurement;

    cout << "Enter a length in inches, and I will convert\n";
    cout << "it to centimeters: ";
    cin >> measurement;
    convert(&measurement);
    cout << fixed << setprecision(4);
    cout << "Value in centimeters: " << measurement << endl;
    return 0;
}
//
// Write the function convert here.
//
```

9.12 Look at the following array definition:

```
const int numbers[SIZE] = { 18, 17, 12, 14 };
```

Suppose we want to pass the array to the function processArray in the following manner:

```
processArray(numbers, SIZE);
```

Which of the following function headers is the correct one for the processArray function?

- A) `void processArray(const int *arr, int size)`
- B) `void processArray(int * const arr, int size)`

9.13 Assume ip is a pointer to an int. Write a statement that will dynamically allocate an integer variable and store its address in ip. Write a statement that will free the memory allocated in the statement you wrote above.

9.14 Assume ip is a pointer to an int. Then, write a statement that will dynamically allocate an array of 500 integers and store its address in ip. Write a statement that will free the memory allocated in the statement you just wrote.

9.15 What is a null pointer?

9.16 Give an example of a function that correctly returns a pointer.

9.17 Give an example of a function that incorrectly returns a pointer.

HỎI ĐÁP



Bài tập

- **Bài tập chương 8:**
 - Chapter8_Programming Challenges.pdf



Trân trọng cảm ơn!

ĐẬU HẢI PHONG

Giảng viên

dauhaiphong@dainam.edu.vn

0912441435