VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Kỹ Thuật Lập Trình (Cơ bản và nâng cao C++)

KTLT1 - HK242

TASK 4 POINT

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 2/2025

1 Trắc Nghiệm

1. Con trỏ là gì trong C++?
   a) Một biến lưu địa chỉ của biến khác.
   b) Một kiểu dữ liệu mới trong C++.
   c) Một loại mảng động.
   d) Một toán tử trong C++.

2. Toán tử nào được sử dụng để lấy địa chỉ của một biến?
   a) `*`
   b) `&`
   c) `#`
   d) `$`

3. Toán tử nào được sử dụng để truy xuất giá trị từ một con trỏ?
   a) `*`
   b) `&`
   c) `@`
   d) `->`

4. Kết quả của biểu thức `int *p = NULL;` là gì?
   a) Con trỏ `p` trỏ đến địa chỉ 0.
   b) Con trỏ `p` có giá trị rác.
   c) Lỗi biên dịch.
   d) Lỗi runtime.

5. Một con trỏ không thể được gán với giá trị nào dưới đây?
   a) Địa chỉ của một biến hợp lệ.
   b) `NULL`.
   c) Một số nguyên bất kỳ.
   d) Kết quả của toán tử `new`.

6. Cách khai báo nào sau đây là đúng?
   a) `int *p;`
   b) `int p*;`
   c) `int* p;`
   d) Cả (1) và (3).

7. Điều gì xảy ra khi một con trỏ trỏ đến vùng nhớ không hợp lệ?
   a) Chương trình hoạt động bình thường.
   b) Lỗi biên dịch.
   c) Lỗi runtime.
   d) Không có gì xảy ra.

8. Để cấp phát động một mảng 10 phần tử kiểu `int`, ta sử dụng lệnh nào?
   a) `int *p = new int[10];`
   b) `int p = new int(10);`
   c) `int p[10];`
   d) `int *p = malloc(10 * sizeof(int));`

9. Câu lệnh nào dùng để giải phóng bộ nhớ động đã cấp phát?
   a) `delete p;`
   b) `free(p);`
   c) `delete[] p;`
   d) Cả (1) và (3).

10. Con trỏ có thể trỏ đến kiểu dữ liệu nào?
    a) Chỉ kiểu `int`.
    b) Chỉ kiểu `char`.
    c) Bất kỳ kiểu dữ liệu nào.
    d) Chỉ kiểu số nguyên.

11. Câu lệnh `p++;` khi `p` là con trỏ kiểu `int` có nghĩa là gì?
    a) Tăng địa chỉ của `p` thêm 1 byte.
    b) Tăng địa chỉ của `p` thêm 4 byte (trên hệ thống 32-bit).
    c) Tăng giá trị mà con trỏ `p` trỏ tới.
    d) Không có tác dụng.

12. Giá trị mặc định của một con trỏ chưa khởi tạo là gì?
    a) `NULL`
    b) Địa chỉ 0x0
    c) Giá trị rác
    d) Không xác định

13. Khi nào nên sử dụng con trỏ?
    a) Khi cần quản lý bộ nhớ động.
    b) Khi cần truyền tham chiếu vào hàm.
    c) Khi làm việc với cấu trúc dữ liệu liên kết.
    d) Cả ba đáp án trên.

14. Để truy xuất thành viên của một struct qua con trỏ, ta dùng toán tử nào?
    a) `.`
    b) `->`
    c) `*`
    d) `&`

15. Một con trỏ NULL có thể được sử dụng để làm gì?

    a) Làm điều kiện kết thúc trong danh sách liên kết.
    b) Kiểm tra con trỏ hợp lệ.
    c) Tránh lỗi truy cập vùng nhớ.
    d) Cả ba đáp án trên.

16. Cho đoạn code sau:

```
int a = 10;
int *p = &a;
```

    Giả sử biến `a` có địa chỉ `0x1000`, vậy giá trị của `p` là gì?

    a) 0x1000
    b) 10
    c) Không xác định
    d) 0x2000

17. Cho đoạn code:

```
int a = 5;
int *p = &a;
int **pp = &p;
```

    Giả sử `a` nằm tại địa chỉ `0x2000` và `p` nằm tại `0x3000`, giá trị của `pp` là gì?

    a) 0x2000
    b) 0x3000
    c) 5
    d) Không xác định

18. Cho đoạn code:

```
int arr[] = {1, 2, 3, 4};
int *p = arr;
```

    Giả sử mảng `arr` bắt đầu tại địa chỉ `0x4000`, vậy `p + 2` trỏ tới địa chỉ nào?

    a) 0x4002
    b) 0x4004
    c) 0x4008
    d) 0x4010

19. Cho đoạn code:

```
struct Point {
    int x;
    int y;
};
struct Point p1 = {3, 4};
struct Point *ptr = &p1;
```

    Giả sử `p1` có địa chỉ `0x5000`, giá trị của `&ptr->y` là gì?

    a) 0x5000
    b) 0x5004
    c) 0x5008
    d) Không xác định

20. Cho đoạn code:

```
char str[] = "Hello";
char *p = str;
```

    Giả sử chuỗi `str` bắt đầu từ `0x6000`, địa chỉ của `p+3` là bao nhiêu?

    a) 0x6003
    b) 0x6006
    c) 0x6009
    d) Không xác định

21. Cho đoạn code:

```
int x = 42;
int *p1 = &x;
int **p2 = &p1;
```

    Giả sử `x` có địa chỉ `0x7000`, `p1` có địa chỉ `0x8000`, vậy `p2` chứa giá trị nào?

    a) 0x7000
    b) 0x8000
    c) 42
    d) Không xác định

22. Cho đoạn code:

```
int arr[3] = {10, 20, 30};
int *p = arr;
```

Giả sử `arr` có địa chỉ `0x9000`, địa chỉ của `p+1` là bao nhiêu?

a) 0x9002 b) 0x9004 c) 0x9008 d) 0x9010

23. Cho đoạn code:

```
struct Data {
    char c;
    int num;
};
struct Data d = {'A', 100};
struct Data *ptr = &d;
```

Giả sử `d` có địa chỉ `0xA000`, địa chỉ của `&ptr->num` có thể là bao nhiêu (giả sử padding)?

a) 0xA001 b) 0xA002 c) 0xA004 d) 0xA008

24. Cho đoạn code:

```
char *s = "Example";
```

Giả sử chuỗi hằng số này nằm tại địa chỉ `0xB000`, vậy giá trị của `s` là gì?

a) 0xB000 b) 0xB002 c) "Example" d) Không xác định

25. Cho đoạn code:

```
int val = 77;
int *p1 = &val;
int **p2 = &p1;
int ***p3 = &p2;
```

Giả sử `val` có địa chỉ `0xC000`, `p1` có địa chỉ `0xD000`, `p2` có địa chỉ `0xE000`, giá trị của `***p3` là gì?

a) 0xC000 b) 0xD000 c) 77 d) Không xác định

26. Kiểu dữ liệu của `p` trong `void *p;` là gì?

a) Con trỏ kiểu `void`.
b) Con trỏ không có kiểu xác định.
c) Con trỏ có thể trỏ đến bất kỳ kiểu dữ liệu nào.
d) Cả (2) và (3).

27. Phép toán nào không hợp lệ với con trỏ?

a) Cộng một số nguyên với con trỏ.
b) Trừ hai con trỏ cùng kiểu.
c) Chia một con trỏ cho một số nguyên.
d) So sánh hai con trỏ.

28. Điều gì xảy ra khi giải phóng một con trỏ hai lần?

a) Không có gì xảy ra.
b) Lỗi biên dịch.
c) Hành vi không xác định.
d) Con trỏ trở về giá trị NULL.

29. Điều gì xảy ra nếu truy cập một con trỏ NULL?

a) Lỗi biên dịch
b) Hành vi không xác định (UB)
c) Trả về giá trị mặc định
d) Gây lỗi phân đoạn ngay lập tức

30. Khi nào một con trỏ "dangling" (con trỏ treo) có thể xuất hiện?

a) Khi con trỏ chưa được khởi tạo
b) Khi vùng nhớ được giải phóng nhưng con trỏ vẫn trỏ đến nó
c) Khi gán con trỏ cho một giá trị hợp lệ
d) Không bao giờ xảy ra trong C++

31. Cách nào giúp tránh rò rỉ bộ nhớ khi sử dụng con trỏ động?

a) Luôn giải phóng bộ nhớ sau khi sử dụng
b) Sử dụng smart pointer như std::unique_ptr hoặc std::shared_ptr

c) Không bao giờ dùng cấp phát động
d) Cả 1 và 2 đều đúng

32. Điều gì xảy ra nếu giải phóng cùng một con trỏ động hai lần?

a) Lỗi biên dịch
b) Hành vi không xác định (UB)
c) Không có lỗi, nhưng vùng nhớ bị mất
d) C++ tự động ngăn chặn điều này

33. Con trỏ void* có thể sử dụng để làm gì?

a) Trỏ đến bất kỳ kiểu dữ liệu nào
b) Thực hiện toán tử số học
c) Truy cập phần tử trực tiếp
d) Không thể sử dụng trong C++

34. Điều gì xảy ra khi dereference một con trỏ chưa được khởi tạo?

a) Chương trình chạy bình thường
b) Hành vi không xác định (UB)
c) Giá trị ngẫu nhiên được trả về
d) Trình biên dịch tự động gán giá trị mặc định

35. Con trỏ hàm trong C++ có tác dụng gì?

a) Trỏ đến một vùng nhớ động
b) Trỏ đến một biến kiểu int
c) Trỏ đến một hàm và có thể gọi hàm đó
d) Không có trong C++

36. new và malloc khác nhau như thế nào?

a) new trả về con trỏ kiểu cụ thể, malloc trả về void*
b) new gọi constructor, malloc không gọi
c) malloc cần free, new cần delete
d) Cả 3 đáp án trên đều đúng

37. Khi nào cần sử dụng delete[] thay vì delete?

a) Khi giải phóng mảng cấp phát động
b) Khi giải phóng con trỏ void*
c) Khi sử dụng smart pointer
d) Không có trường hợp nào cần dùng delete[]

2 Đọc Code

Câu 1. Kết quả của chương trình

```
int a = 42;
int *p = &a;
int **q = &p;
int ***r = &q;

cout << "a = " << a << endl;
cout << "&a = " << &a << endl;
cout << "p = " << p << ", *p = " << *p << endl;
cout << "&p = " << &p << endl;
cout << "q = " << q << ", *q = " << *q << ", **q = " << **q << endl;
cout << "&q = " << &q << endl;
cout << "r = " << r << ", *r = " << *r << ", **r = " << **r << ", ***r = " << ***r << endl;
cout << "&r = " << &r << endl;

***r = 99;
cout << "After ***r = 99, a = " << a << endl;
```

Kết quả và giải thích: ...

- ...
- ...

Câu 2. Kết quả của chương trình

```
int arr[] = {10, 20, 30, 40, 50};
int *ptr = arr;

cout << "Initial values:" << endl;
cout << "*ptr = " << *ptr << endl;
cout << "*(ptr + 1) = " << *(ptr + 1) << endl;
cout << "*(ptr + 2) = " << *(ptr + 2) << endl;

cout << "\nUsing ptr++:" << endl;
cout << "*ptr++ = " << *ptr++ << endl;
cout << "*ptr = " << *ptr << endl;

cout << "\nUsing ++*ptr:" << endl;
++*ptr;
cout << "*ptr = " << *ptr << endl;

cout << "\nUsing (*ptr)++:" << endl;
(*ptr)++;
cout << "*ptr = " << *ptr << endl;

cout << "\nUsing --ptr:" << endl;
--ptr;
cout << "*ptr = " << *ptr << endl;

```

Kết quả và giải thích: ...

- ...
- ...

Câu 3. Kết quả của chương trình

```
int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
int (*ptr)[3] = arr;

cout << "Using array indexing:" << endl;
cout << arr[0][1] << " " << arr[1][2] <<
 ↪  endl;

cout << "\nUsing pointer notation:" << endl;
cout << *(*(arr + 0) + 1) << " " << *(*(arr +
 ↪  1) + 2) << endl;

cout << "\nUsing pointer ptr++:" << endl;
cout << (*ptr)[1] << " ";
ptr++;
cout << (*ptr)[1] << endl;
```

Kết quả và giải thích: ...

- ...
- ...

Câu 4. Kết quả của chương trình

```
int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
int (*ptr)[3] = arr;

cout << "Address of array:" << endl;
cout << "arr       : " << arr << endl;
cout << "arr[0]    : " << arr[0] << endl;
cout << "arr[1]    : " << arr[1] << endl;

cout << "\nAddress of elements:" << endl;
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
        cout << &arr[i][j] << endl;
    }
}

cout << "\nUsing pointer ptr:" << endl;
cout << "ptr       : " << ptr << endl;
cout << "*ptr      : " << *ptr << endl;
ptr++;
cout << "ptr++     : " << ptr << endl;
cout << "*ptr      : " << *ptr << endl;
```

Kết quả và giải thích: ...

- ...
- ...

Câu 5. Kết quả của chương trình

Kết quả và giải thích: ...

- ...
- ...

```cpp
struct Point {
    int x, y;
};

Point p1 = {10, 20};
Point *ptr = &p1;

cout << "Address of struct:" << endl;
cout << "&p1         : " << &p1 << endl;
cout << "&p1.x       : " << &p1.x << endl;
cout << "&p1.y       : " << &p1.y << endl;

cout << "\nValues using pointer:" << endl;
cout << "ptr->x      : " << ptr->x << endl;
cout << "ptr->y      : " << ptr->y << endl;

ptr->x += 5;
ptr->y -= 5;

cout << "\nUpdated values:" << endl;
cout << "p1.x        : " << p1.x << endl;
cout << "p1.y        : " << p1.y << endl;
```

Câu 6. Kết quả của chương trình

Kết quả và giải thích: ...

- ...
- ...

```cpp
void modifyValue(int *ptr) {
    *ptr += 10;
    ptr = new int(10);
}

void modifyPointer(int *&ptr) {
    *ptr += 10;
    ptr = new int(10);
}

int a = 20;
int *p = &a;
cout << "Before modifyValue: " << *p << endl;
modifyValue(p);
cout << "After modifyValue: " << *p << endl;

modifyPointer(p);
cout << "After modifyPointer: " << *p <<
 ↪  endl;

cout << "a: " << a << endl;
```

Câu 7. Kết quả của chương trình

Kết quả và giải thích: ...

- ...
- ...

```
void modifyValue(int **ptr) {
    **ptr += 10;
    ptr = new int*[2];
}

void modifyPointer(int **&ptr) {
    **ptr += 10;
    ptr = new int*[2];
    for (int i = 0; i < 2; i++) {
        ptr[i] = new int[2]{50, 60};
    }
}

int arr[2][2] = {{1, 2}, {3, 4}};
int *pArr[2] = {arr[0], arr[1]};
int **ptr = pArr;

cout << "Before modifyValue: " << **ptr <<
 ↪  endl;
modifyValue(ptr);
cout << "After modifyValue: " << **ptr <<
 ↪  endl;

modifyPointer(ptr);
cout << "After modifyPointer: " << **ptr <<
 ↪  endl;
cout << "New Array[0][0]: " << ptr[0][0] <<
 ↪  ", New Array[0][1]: " << ptr[0][1] <<
 ↪  endl;
```

Câu 8. Kết quả của chương trình

Kết quả và giải thích: ...

- ...
- ...

```
void modifyValue(int (*ptr)[]) {
    (*ptr)[0] += 10;
}

void modifyPointer(int (*&ptr)[]) {
    ptr = new int[2][2]{{50, 60}, {70, 80}};
}

int arr[2][2] = {{1, 2}, {3, 4}};
int (*ptr)[2] = arr;

cout << "Before modifyValue: " << ptr[0][0]
 ↪  << endl;
modifyValue(ptr);
cout << "After modifyValue: " << ptr[0][0] <<
 ↪  endl;

modifyPointer(ptr);
cout << "After modifyPointer: " << ptr[0][0]
 ↪  << ", " << ptr[0][1] << endl;
```

3 Bài Tập

Câu 1: Sử dụng con trỏ để đảo ngược mảng

Đề bài: Viết hàm sử dụng con trỏ để đảo ngược một mảng số nguyên.

```
void reverseArray(int *arr, int n) {
    // TODO
}
```

Test case

Input	Output
1 2 3 4 5	5 4 3 2 1
10 20 30 40	40 30 20 10
7 8 9	9 8 7
42	42

Câu 2: Sắp xếp mảng sinh viên theo điểm số

Đề bài: Viết chương trình sử dụng con trỏ để sắp xếp danh sách sinh viên theo điểm số theo thứ tự giảm dần.

```
struct Student {
    string name;
    float score;
};

void sortStudents(Student *arr, int n) {
    // TODO: Sắp xếp sinh viên theo điểm giảm dần
}
```

Test case

Input	Output
"A", 8.5, "B", 9.2, "C", 7.8	"B", 9.2, "A", 8.5, "C", 7.8
"X", 6.0, "Y", 8.0, "Z", 7.5	"Y", 8.0, "Z", 7.5, "X", 6.0

Câu 3: Tính tổng từng hàng của mảng 2 chiều

Đề bài: Viết hàm nhận vào một mảng 2 chiều số nguyên và trả về con trỏ trỏ đến mảng 1 chiều chứa tổng từng hàng của mảng.

```
int* rowSum(int arr[][4], int rows) {
    // TODO: Tính tổng từng hàng và trả về con trỏ
}
```

Test case

Input	Output
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12	10, 26, 42
2, 4, 6, 8, 1, 3, 5, 7, 9, 0, 11, 2	20, 16, 22

Câu 4: Nhân hai ma trận và trả về mảng 2 chiều

Đề bài: Viết hàm nhận vào hai ma trận 2 chiều (kích thước cố định 3x3) và trả về ma trận kết quả dưới dạng con trỏ.

```
int** multiplyMatrix(int A[3][3], int B[3][3]) {
    // TODO
}

void printMatrix(const int** &matrix) {
    // TODO
}
```

Test case

Input	Output
$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ $B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$	$\text{Result} = \begin{bmatrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{bmatrix}$

Câu 5: Biến đổi ma trận xoắn ốc bằng con trỏ

Đề bài: Viết hàm nhận vào một ma trận số nguyên động kích thước $N \times M$, biến đổi nó theo thứ tự xoắn ốc và trả về một mảng động 1 chiều chứa các phần tử theo thứ tự xoắn ốc.

```
int* spiralOrder(int** matrix, int rows, int cols) {
    // TODO
}

void printArray(const int* &arr, int size) {
    // TODO
}
```

Test case

Input	Output
Ma trận: $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$	Mảng kết quả: $[1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]$