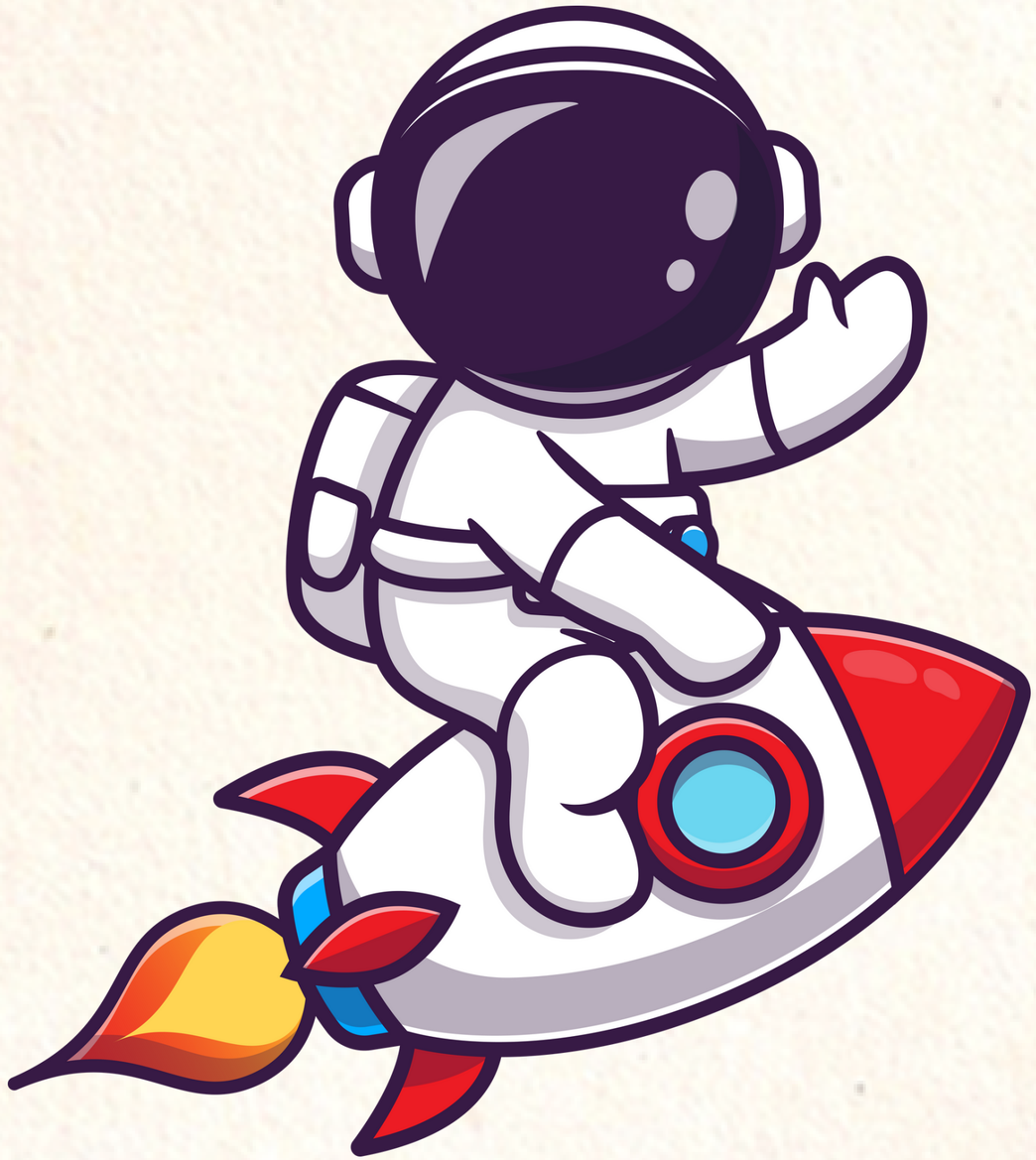


Phitron

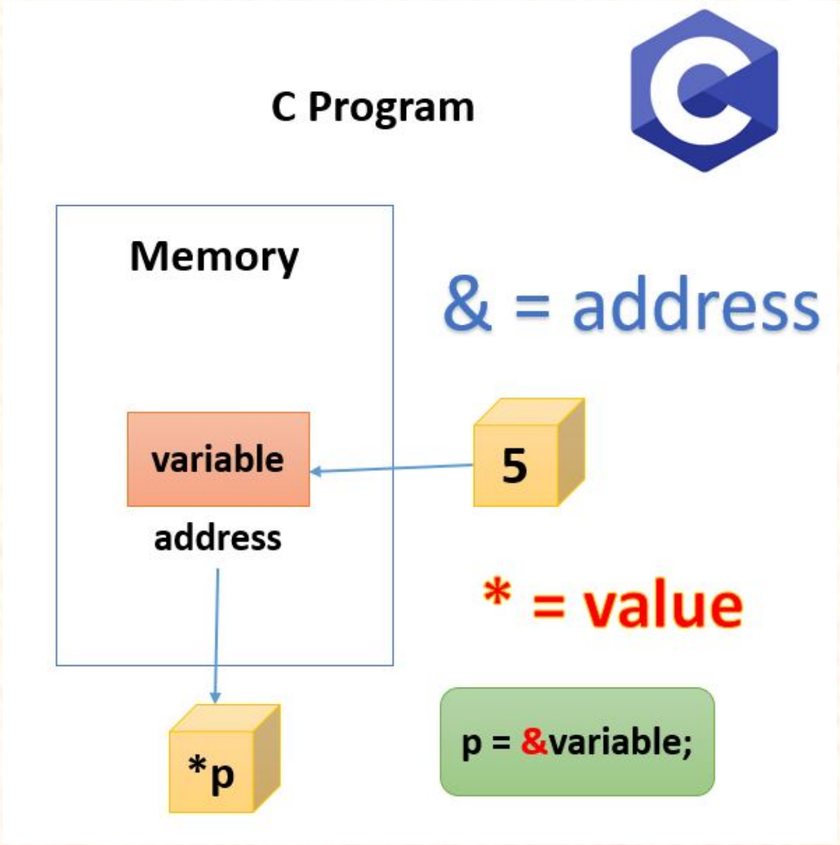
Outlines

- Pointer
- functions
- Exam Hacks

Phitron

Pointers

A pointer is a variable that stores the memory address of another variable as its value.



Phitron

Pointers

In How Many Ways Can You Access Variables?

- **Direct Access: We can directly use the variable name to access the variable.**
- **Indirect Access: We use a pointer to access that variable.**

```c
int a = 5, *ptr;
ptr = &a;
printf("Direct Access, a = % d\n", a);

printf("Indirect Access, a = % d\n", *ptr);
```

Phitron

Pointers

Why indirect access of values is important

- **Dynamic Memory allocation**
- **Passing parameters by reference**
- **Accessing Heap memory**
- **Wifi access**
- **Monitor access**
- **Keyboard access**

Pointers

Reference :

A reference is a memory address that points to the location of a variable or object in memory. In C, a reference is represented by a pointer variable.

Doesn't consume extra memory. address and value of a and p are same.

```c
int a = 10;
int *p;
p = &a; // p points to the memory address of a

printf("The value of a is %d\n", a);
printf("The value of p is %p\n", p);
```

```
a ---> Data Variable
*a ---> pointer
&a ----> reference
```

Phitron

Pointers

DeReference :

Dereferencing is the process of accessing the value stored in the memory address pointed to by a pointer variable.

```c
int a = 10;
int *p;
p = &a; // p points to the memory address of a

printf("The value of a is %d\n", a);
printf("The value of p is %p\n", p);
printf("The value of p is %d\n", *p);
```

```
a ---> Data Variable
*a ---> pointer
&a ----> reference
*p ----> Dereference
```

Phitron

Pointers

What Are the Use Cases of Pointers in C?

- **int/float/char/double pointers**
- **Pointer arithmetic**
- **Pointer to pointer**
- **Array of pointers**
- **Call by value**
- **Call by reference**

Phitron

Pointers

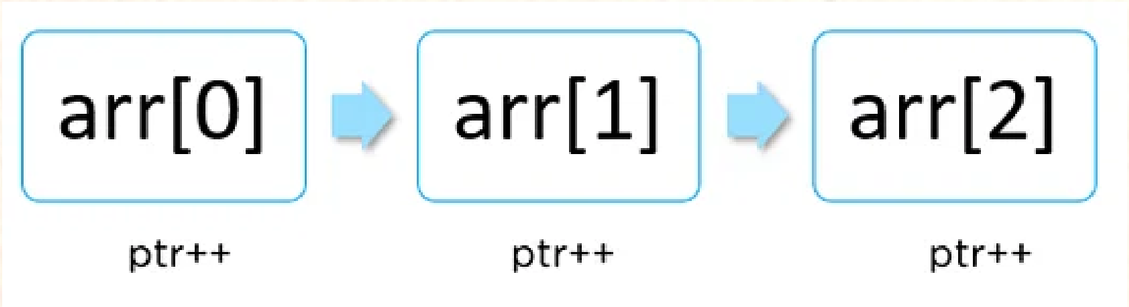
int pointer :

```c
int age = 43;     // An int variable
int *ptr = &age; // A pointer variable, with the name ptr, that stores the address of age

// Output the value of age (43)
printf("%d\n", age);

// Output the memory address of age
printf("%p\n", &age);

// Output the memory address of age with the pointer
printf("%p\n", ptr);
```

Phitron

Pointers

Pointer arithmetic :




```c
int arr[3] = {50, 150, 200};
int *ptr;
ptr = arr;
for (int i = 0; i < 3; i++)
{
    printf("Value of *ptr = % d\n", *ptr);
    printf("Address of *ptr = % d\n", ptr);
    ptr++;
}
```

```
Value of *ptr =  50
Address of *ptr =  6422284
Value of *ptr =  150
Address of *ptr =  6422288
Value of *ptr =  200
Address of *ptr =  6422292
```

Phitron

Pointers

Array of pointers :

```c
int arr[3] = {50, 150, 200};
int *ptr;
ptr = arr; // ptr = &arr[0];
printf("%d\n", ptr);
printf("%d\n", ptr[0]);
printf("%d\n", ptr[1]);
printf("%d\n", ptr[2]);
```

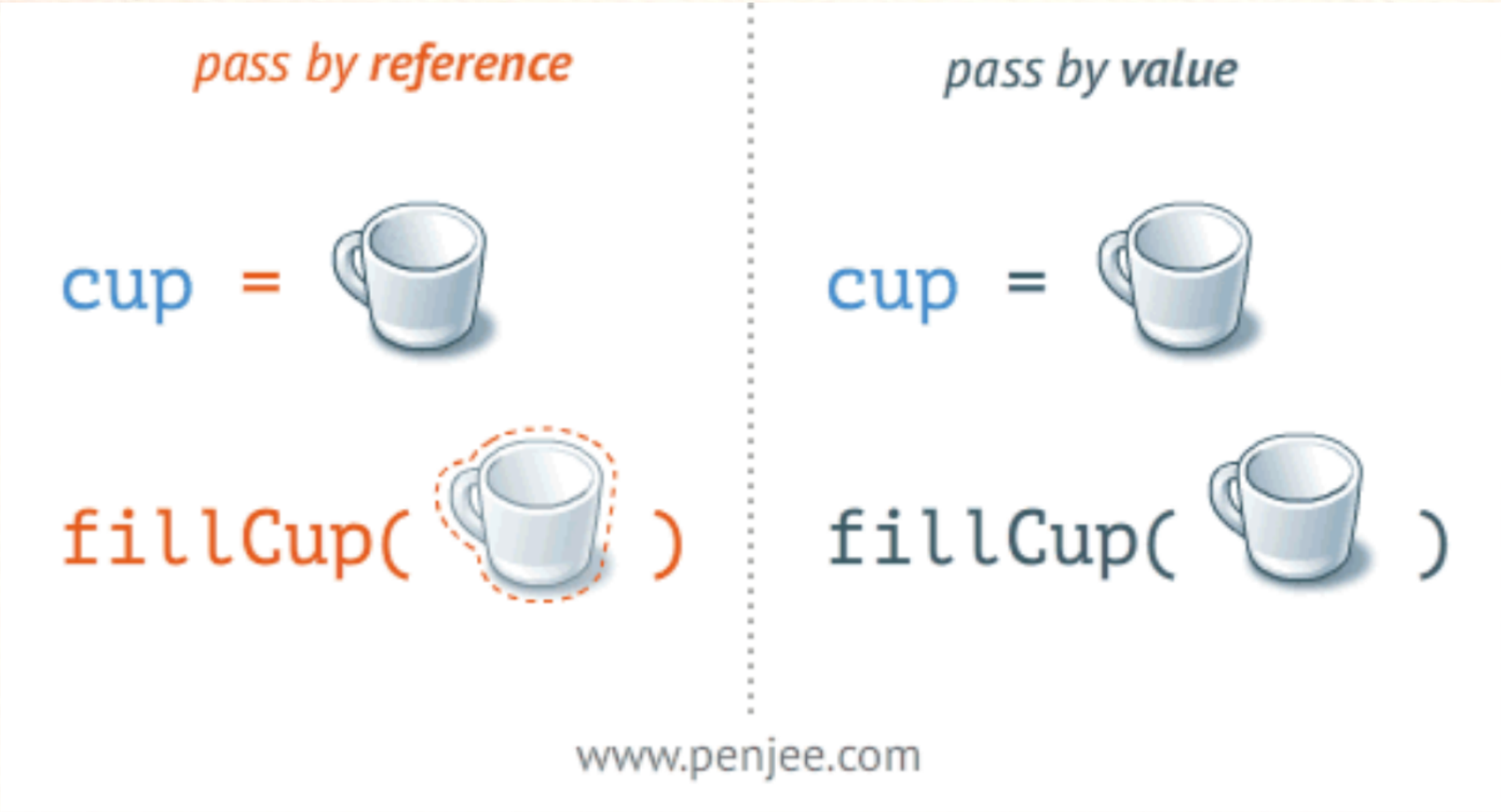
```
6422288
50
150
200
```

Phitron

Pointers

pointers in function :

animation link : https://blog.penjee.com/wp-content/uploads/2015/02/pass-by-reference-vs-pass-by-value-animation.gif



Phitron

Pointers

pointers in function :

```c
#include <stdio.h>

void increment(int x)
{
    x = x + 1;
    printf("Inside function: x = %d\n", x);
}
int main()
{
    int x = 5;
    printf("Before function call: x = %d\n", x);
    increment(x);
    printf("After function call: x = %d\n", x);
    return 0;
}
```

call by value

```c
#include <stdio.h>

void increment(int *x) {
    *x = *x + 1;
    printf("Inside function: x = %d\n", *x);
}

int main() {
    int x = 5;
    printf("Before function call: x = %d\n", x);
    increment(&x);
    printf("After function call: x = %d\n", x);
    return 0;
}
```

call by reference

Phitron

Pointers

pointers in function :

```c
#include <stdio.h>
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main()
{
    int a = 10, b = 15;
    swap(&a, &b);
    printf("%d %d", a, b);
    return 0;
}
```

Phitron

Phitron