

# Memory-Related Perils and Pitfalls

- Dereferencing bad pointers
- Reading uninitialized memory
- Overwriting memory
- Referencing nonexistent variables
- Freeing blocks multiple times
- Referencing freed blocks
- Failing to free blocks

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# C operators

## Operators

`() [] -> .`

`! ~ ++ -- + - * & (type) sizeof`

`* / %`

`+ -`

`<< >>`

`< <= > >=`

`== !=`

`&`

`^`

`|`

`&&`

`||`

`?:`

`= += -= *= /= %= &= ^= != <<= >>=`

`,`

## Associativity

left to right

right to left

left to right

left to right

left to right

left to right

left to right

left to right

left to right

left to right

left to right

left to right

right to left

right to left

left to right

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- `->`, `()`, and `[]` have high precedence, with `*` and `&` just below
- Unary `+`, `-`, and `*` have higher precedence than binary forms

# C Pointer Declarations: Test Yourself!

```
int *p
```

p is a pointer to int

```
int *p[13]
```

p is an array[13] of pointer to int

```
int *(p[13])
```

p is an array[13] of pointer to int

```
int **p
```

p is a pointer to a pointer to an int

```
int (*p)[13]
```

p is a pointer to an array[13] of int

```
int *f()
```

f is a function returning a pointer to int

```
int (*f)()
```

f is a pointer to a function returning int

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Dereferencing Bad Pointers

## ■ The classic scanf bug

```
int val;
```

```
...
```

```
scanf("%d", val);
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Reading Uninitialized Memory

- Assuming that heap data is initialized to zero

```
/* return y = Ax */
int *matvec(int **A, int *x) {
    int *y = malloc(N*sizeof(int));
    int i, j;

    for (i=0; i<N; i++)
        for (j=0; j<N; j++)
            y[i] += A[i][j]*x[j];
    return y;
}
```

Assignment Project Exam Help  
<https://powcoder.com>  
Add WeChat powcoder

# Overwriting Memory

- Allocating the (possibly) wrong sized object

```
int **p;
```

```
p = malloc(N*sizeof(int));
```

```
for (i=0; i<N; i++) {
```

```
    p[i] = malloc(M*sizeof(int));
```

```
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Overwriting Memory

## ■ Off-by-one error

```
int **p;  
  
p = malloc(N*sizeof(int *));  
  
for (i=0; i<=N; i++) {  
    p[i] = malloc(M*sizeof(int));  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Overwriting Memory

- Not checking the max string size

```
char s[8];
```

```
int i;
```

```
gets(s); /* reads "123456789" from stdin */
```

Assignment Project Exam Help

<https://powcoder.com>

- Basis for classic buffer overflow attacks

Add WeChat powcoder



# Overwriting Memory

## ■ Misunderstanding pointer arithmetic

```
int *search(int *p, int val) {  
    while (*p && *p != val)  
        p += sizeof(int);  
    return p;  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder