POINTER OPERATIONS

Arithmetic With Pointers

Delroy A. Brinkerhoff

RELATIONAL OPERATIONS

```
int*        i1;
int*        i2;
Person*     p1;
Person*     p2;

if (i1 == i2) . . . .
if (i1 != i2) . . . .
if (p1 == p2) . . . .
if (p1 != p2) . . . .
```

```
int*        i1 = nullptr;
int*        i2 = 0;     //zero
Person*     p1 = nullptr;
Person*     p2 = 0;     //zero

while (i1 == nullptr) . . . .
while (i2 != 0) . . . .
while (p1 == nullptr) . . . .
while (p2 != 0) . . . .
```

- nullptr replaces NULL and 0

SECURE PROGRAMMING

- Important pointer facts:
  - Pointer variables are not automatically initialized
  - Using a null pointer causes an error: e.g., you can't access a member if the pointer doesn't point to an object
  - Using uninitialized pointers causes difficult to find errors and are a security threat

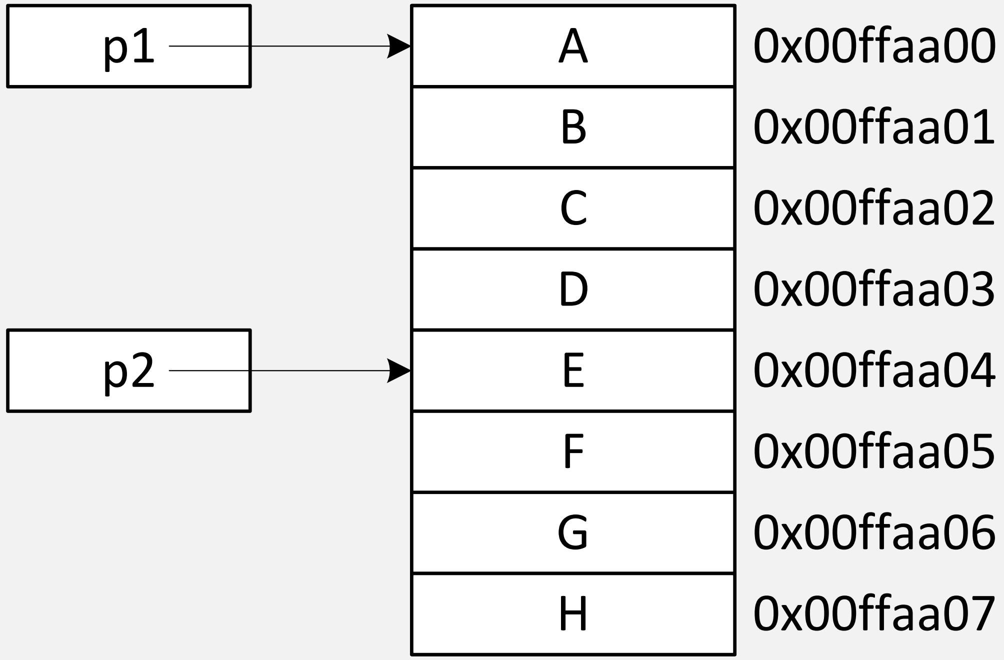
- Steps to minimize errors and enhance security:
  - Initialize pointers: Person* p = nullptr;
  - Test the value stored in a pointer before using it – the underlying problem determines the test and the action
  - if (p == nullptr)
  - if (p != nullptr)

POINTER ARITHMETIC, PART 1

```
char data[] = {
    'A', 'B', 'C', 'D',
    'E', 'F', 'G', 'H'
};

char* p1 = data;
char* p2 = p1 + 4;

p2 points to 'E'
```

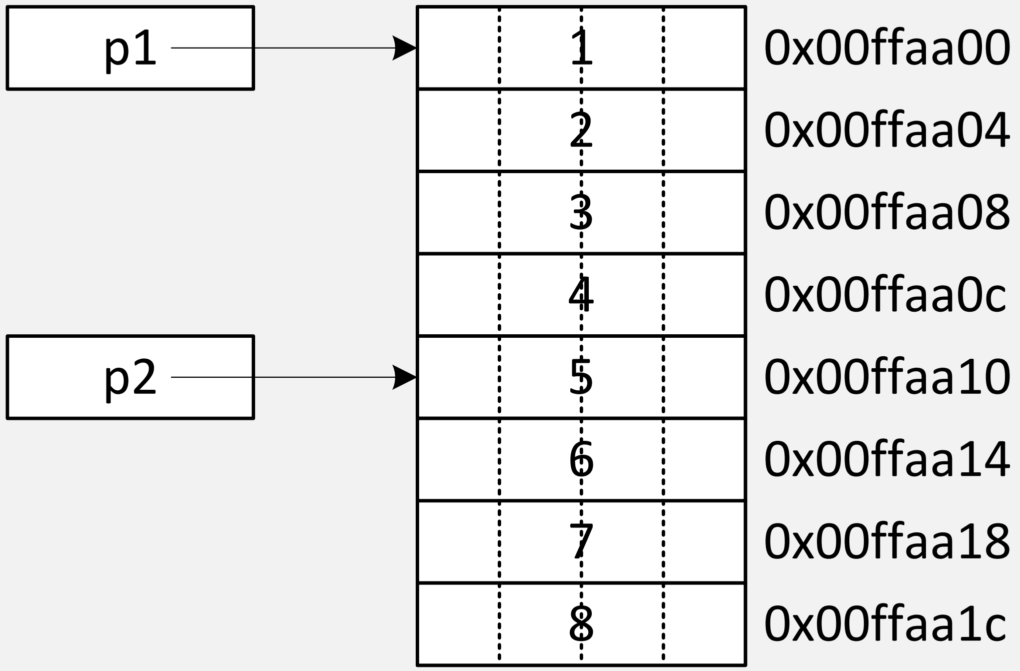


POINTER ARITHMETIC, PART 2

```
int data[] = {
    1, 2, 3, 4, 5, 6, 7, 8
};

int* p1 = data;
int* p2 = p1 + 4;

p2 points to 5
```

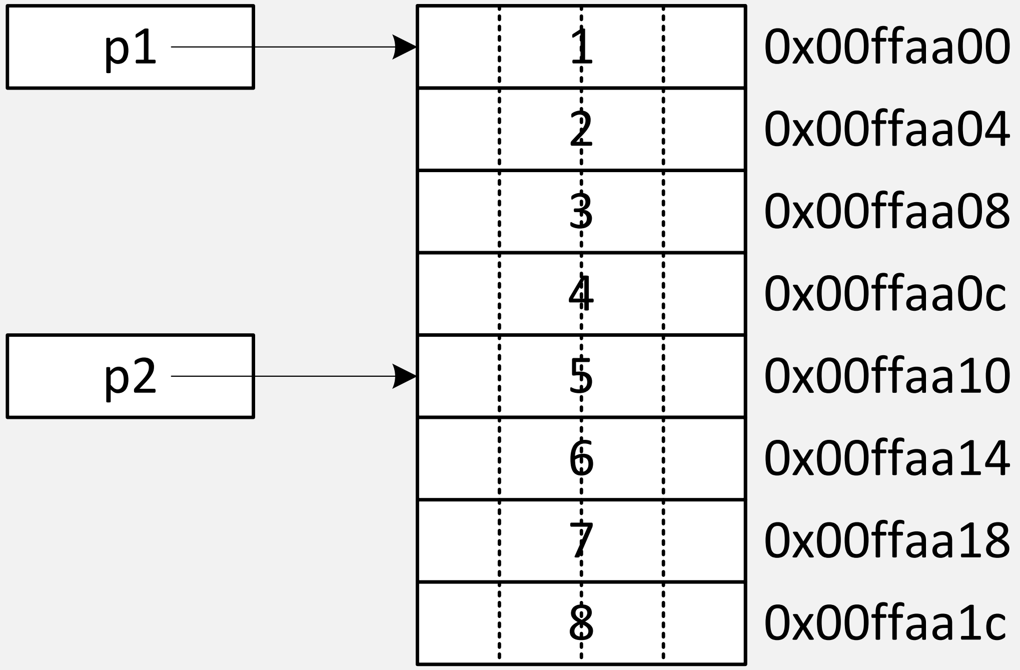


POINTER ARITHMETIC, PART 3

```
int data[] = {
    1, 2, 3, 4, 5, 6, 7, 8
};

int* p1 = data;
int* p2 = p1 + 4;


p2 - p1 is 4
```


p1
p2
1
2
3
4
5
6
7
8
0x00ffaa00
0x00ffaa04
0x00ffaa08
0x00ffaa0c
0x00ffaa10
0x00ffaa14
0x00ffaa18
0x00ffaa1c