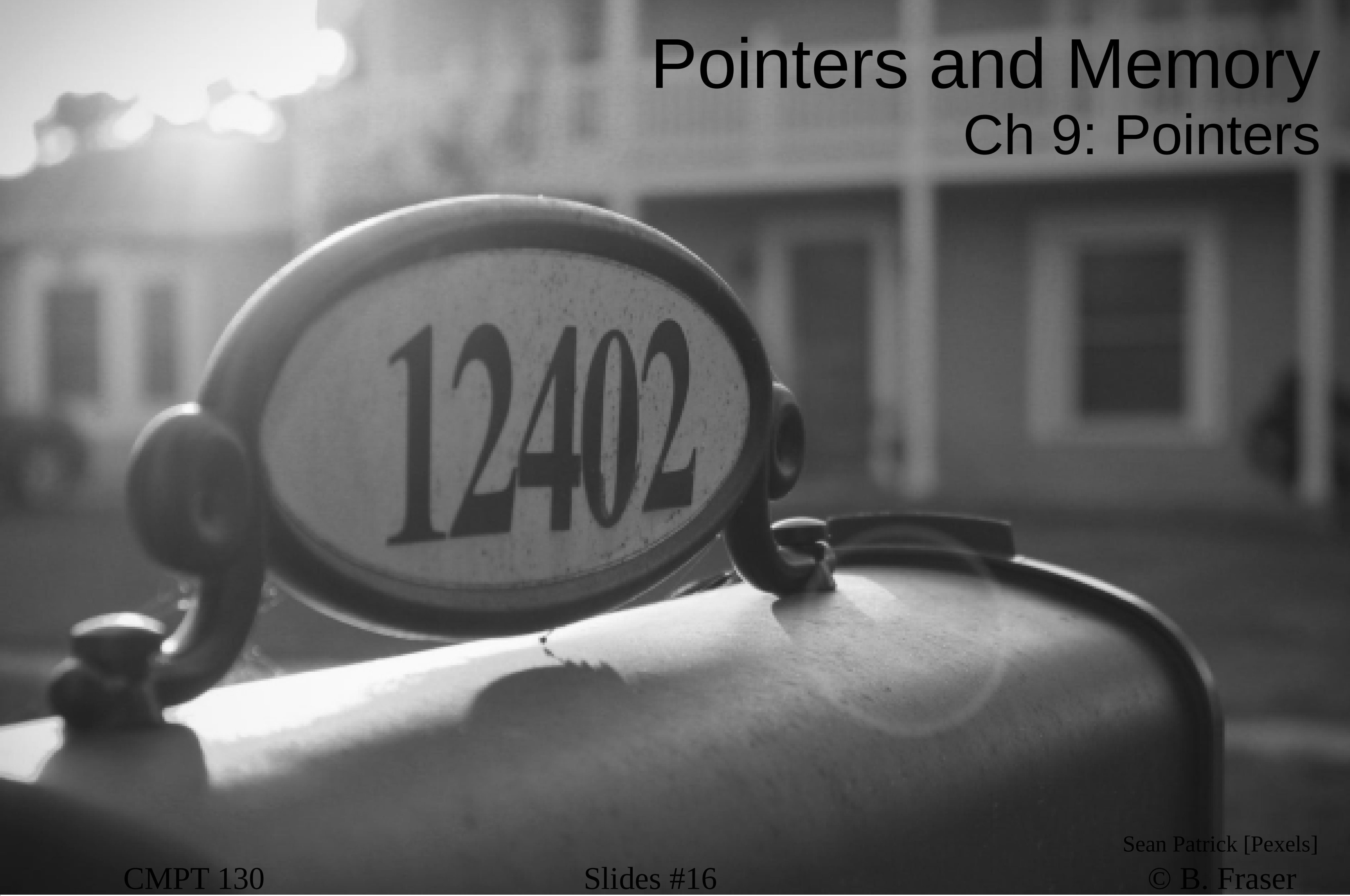


Pointers and Memory

Ch 9: Pointers

Sean Patrick [Pexels]

Topics

1) How can a program work with addresses?
2) How can we use pointers with functions?

Pointers

Address Of

- Variables in Memory
  - Each variable is stored in..
  - Get the address of a variable with..

```
int answer = 42;
cout << "Value:     " << answer << endl;
cout << "Address:   " << &answer << endl;
```

```
Value:      42
Address:    0x7fffe3bd9fac
```

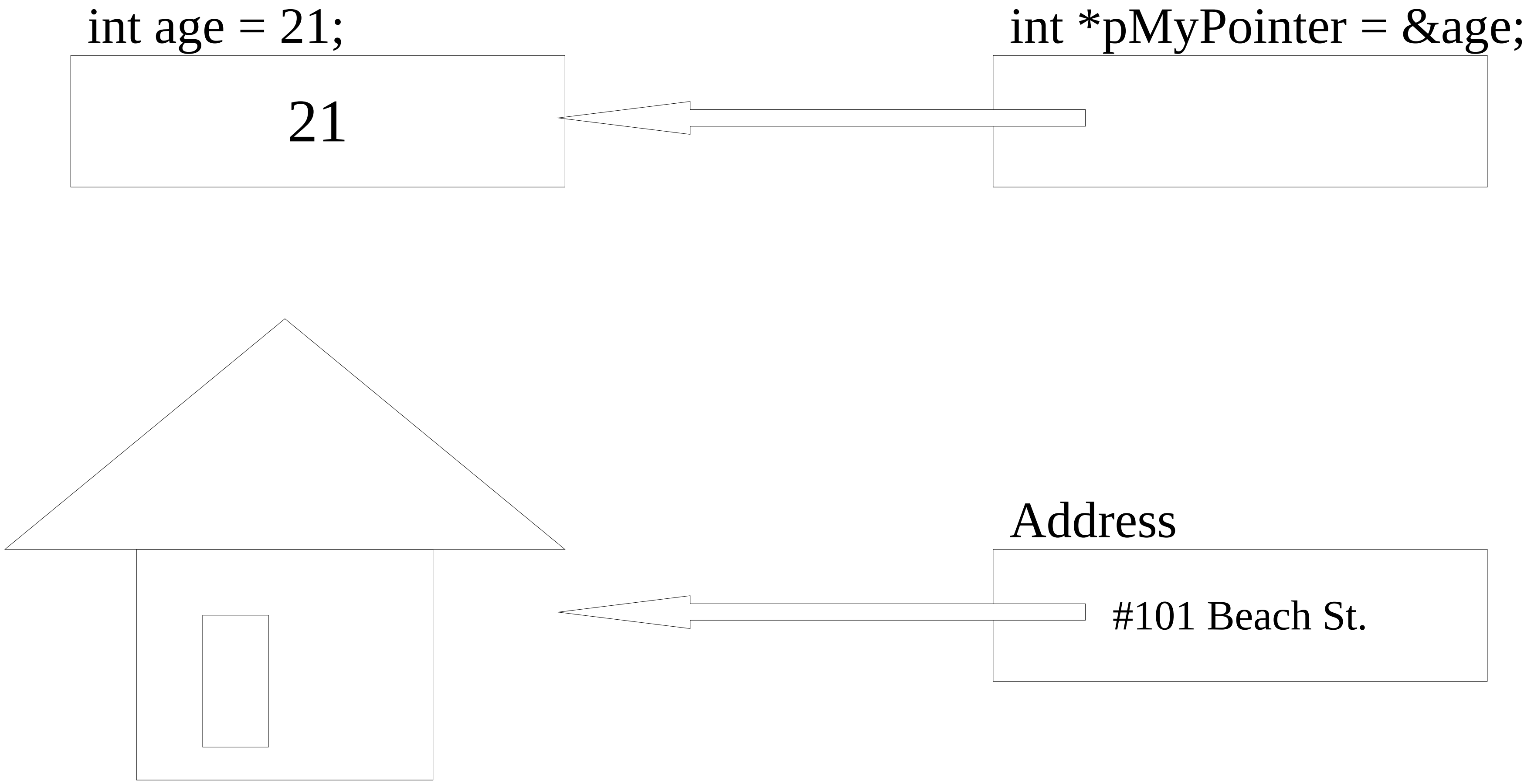
- Can a program work with addresses?

  ..

Pointers

- Pointer
  - Declare a pointer by adding a star after the type:
    int* pStudentNum;
  - You should always initialize the pointer:
    float* pHeight = nullptr;   // Point to nothing.
    int answer = 42;
    int* pAnswer = &answer;
  - All pointers are the same size. ..

Pointers are like House Addresses



Using a Pointer

- Dereference
  -
  - Add a * before the variable name.
- Example

```
int answer = 1;
int* pAns = nullptr;

pAns = &answer;
*pAns = 23;

cout << "Answer: " << answer << endl;
cout << "*pAns:  " << *pAns << endl;
```

```
Answer: 23
*pAns:  23
```

Example

```
// Create a variable
float myPi = 3.14;
cout << "@1: myPi = " << myPi << endl;

// Create pointer; point to the variable.
float* pPi = &myPi;
cout << "@2: pPi  = " << pPi << endl;
cout << "@2: &myPi= " << &myPi << endl;
cout << "@2: *pPi = " << *pPi << endl;

// Change via the variable
myPi = 13.9;
cout << "@3: myPi = " << myPi << endl;
cout << "@3: *pPi = " << *pPi << endl;

// Change via the pointer
*pPi = 999.2;
cout << "@4: myPi = " << myPi << endl;
cout << "@4: *pPi = " << *pPi << endl;
```

..

```
@1: myPi = 3.14
```

```
@2: pPi  = 0x7fff32b8ee44
@2: &myPi= 0x7fff32b8ee44
@2: *pPi = 3.14
```

```
@3: myPi = 13.9
@3: *pPi = 13.9
```

```
@4: myPi = 999.2
@4: *pPi = 999.2
```

Pointer Operator Recap

- Uses of *
  - Multiplication — `int x = 10 * 2;`
  - Pointer declaration — `int *ptr = nullptr;`
  - Pointer dereferencing — `*ptr = (*ptr) / 2 + 1;`
- Uses of &
  - AND — `if (x > 1 && x < 10) {..}`
  - Memory Address — `ptr = &x;`
  - Pass by reference — `int foo(string &str);`

Address	2047	2048	2049	2050	2051	2052	2053
Data							

Working with Pointers

Working with Variables

Review

```
// Complete questions a) through d)
void bustedCode()
{
    int age = 75;
    int* pPointer = nullptr;

    // a) What's wrong with this?
    *pPointer = &age;

    // b) What's wrong with this?
    pPointer = 42;

    // c) Change the value to which pPointer points to zero:


    // d) Change pPointer to point to zero:

}
```

Pass by Pointer

Pass by...

- Pass-by-value
  - Copies of the arguments are passed to the function.
- Pass-by-reference
  - Function works on actual argument (reference).
- Pass-by-pointer
  - ..
    is passed to function.
  - Changes in the function to that location in memory..
  - Pass-by-reference is similar to pass-by-pointer, but it handles the pointer access for you.

Examples

```
void swapByVal(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}



void swapByRef(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}


void swapByPtr(int* pA, int* pB) {
    int temp = *pA;
    *pA = *pB;
    *pB = temp;
}
```

```
int main()
{
    {
        int mine = 1;
        int yours = 99;
        swapByVal(mine, yours);
        cout << mine << " " << yours << endl;
    }

    {
        int mine = 1;
        int yours = 99;
        swapByRef(mine, yours);
        cout << mine << " " << yours << endl;
    }

    {
        int mine = 1;
        int yours = 99;
        swapByPtr(&mine, &yours);
        cout << mine << " " << yours << endl;
    }
}
```

Review

- Write a void function named sort() which:
  - takes two pointers to float's (pX and pY).
  - if *pX > *pY, then swap their values.

```
// Sort calling example:
float one = 100;
float two = 12.5;
sort(&one, &two);
cout << "One = " << one
        << ", Two = " << two
        << endl;
```

```
One = 12.5
Two = 100
```

Summary

- Pointers point to memory locations.
  - Declare: int* pHeight = nullptr;
  - Set: pHeight = &someVariable;
  - Use: cout << *pHeight;
- Pass-by-pointer allows changes to the arguments.