# MODEL OF MEMORY
# C++ ARRAYS

Problem Solving with Computers-I


# Memory and C++ programs

*"The overwhelming majority of program bugs and computer crashes stem from problems of memory access... Such memory-related problems are also notoriously difficult to debug. Yet the role that memory plays in C and C++ programming is a subject often overlooked…. Most professional programmers learn about memory entirely through experience of the trouble it causes."*

…. Frantisek Franek
(Memory as a programming concept)

# Model of memory

- Sequence of adjacent cells
- Each cell has 1-byte stored in it
- Each cell has an address (memory location)

```
char x = 1;
int y = 4;
char tmp = x;
x = y;
y = tmp;
```

# Array motivation

- Write a program to record the midterm scores of 10 students in CS16, by asking the user to input each score. Then print out each of the recorded scores

# C++ Arrays

A C++ array is a **list of elements** that share the same name, have the same data type and are located adjacent to each other in memory

**scores**

| |
|---|
| 10 |
| 20 |
| 30 |
| 40 |
| 50 |
| 60 |
| 70 |
| 80 |

| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 |
|---|---|---|---|---|---|---|---|

**scores**

# What is the memory location of each element?


```
int scores[5]={10, 20, 30, 40, 50};
```

If the starting location of the array is 0x200, what is memory location of element at index 2?

A. 0x201

B. 0x202

C. 0x204

D. 0x208

# Declaring C++ arrays


```
int scores[5];
```

// declares a 5-element integer array

//declare a 5-element char array

# Declaring and initializing, accessing elements

// Declare a 5-element integer array and fill it with values

```
int scores[5]={10, 20, 30, 40, 50};
```

# Exercise: Reassign each value to 60


```
int scores[]={20,10,50}; // declare and initialize
//Access each element and reassign its value to 60
```

# Exercise: Increment each element by 10

```
int scores[]={20,10,50}; // declare and initialize
//Increment each element by 10
```

# C++ 11 range based for loop

```
int scores[]={20,10,50}; // declare and initialize
//Print each element using a range based for loop
```

# Most common array pitfall- out of bound access

```
int arr[]={20,10,50}; // declare and initialize
for(int i=0; i<=3; i++)
    scores[i] = scores[i]+10;
```

# Tracing code involving arrays


```
int arr[]={1,2,3};
int tmp = arr[0];
arr[0] = arr[2];
arr[2] = tmp;
```

Choose the resulting array after the code is executed

**A.**


**B.**


**C.**


**D.** None of the above

# Arrays – motivating example

**DEMO:** Write a program to store 10 scores and calculate the average of the 10 scores.

# POINTERS


Problem Solving with Computers-I


# How comfortable do you feel with using github?

A. Very comfortable in the context of labs; I have a basic understanding of how git works
B. I know how to use it but I have no idea how git works
C. I don’t feel comfortable using it
D. I am completely lost

# Swap function

```
#include <iostream>
using namespace std;
void swap(int a, int b){
        cout<< "Inside swap"<<endl;
        int tmp = a;
        a = b;
        b = tmp;
        cout<< a << "  " << b<< endl;
}

int main(){
        int x= 10, y=20;
        cout<< "Before swap" <<endl;
        cout<< x<< "  " <<y<<endl;
        swap(x, y);
        cout<< "After swap" <<endl;
        cout<< x<< "  " <<y<<endl;
}
```

# Pointers

- Pointer: A variable that contains the address of another variable
- Declaration: *type* * pointer_name;

```
int *p;
```

How do we initialize a pointer?

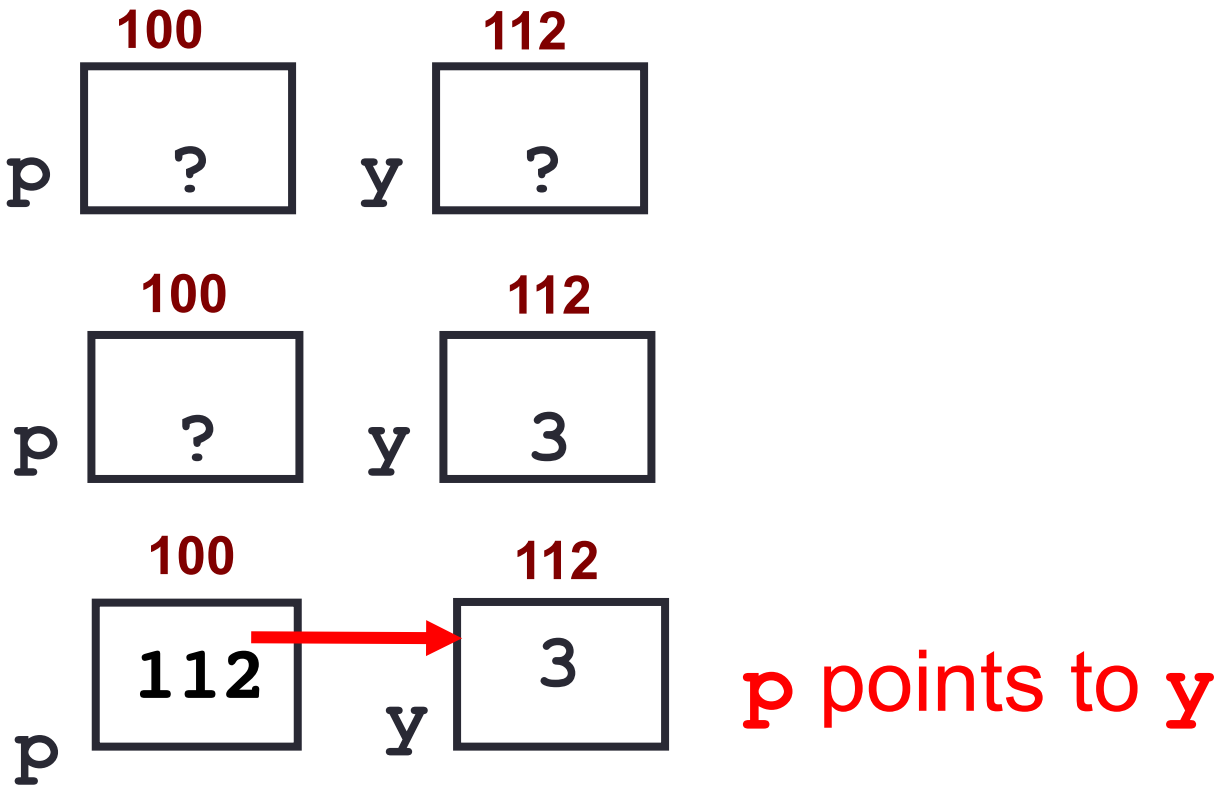
# How to make a pointer point to something

```
int *p;
int y;
```


To access the location of a variable, use the address operator ‘&’

# How to make a pointer point to something

```
int *p, y;
```




**p** points to **y**

# Pointer Diagrams: Diagrams that show the relationship between pointers and pointees

# You can change the value of a variable using a pointer !

```
int *p, y;
y = 3;
p = &y;

*p = 5;
```

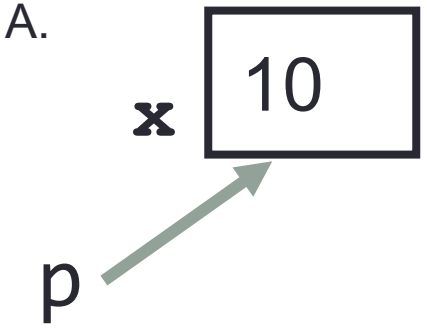
Use dereference * operator to left of pointer name

# Tracing code involving pointers

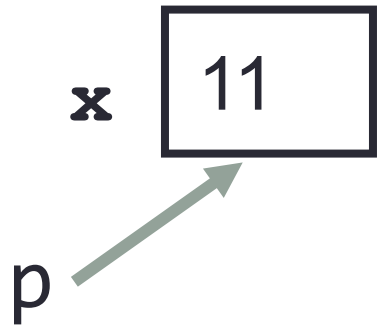
```
int *p, x=10;
p = &x;
*p = *p + 1;
```

Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.

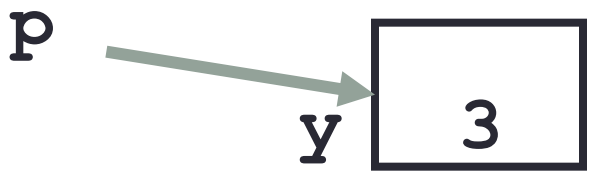



B.




C. Neither, the code is incorrect

# Two ways of changing the value of a variable




Change the value of y directly.

Change the value of y indirectly (via pointer p).

# Pointer assignment and pointer arithmetic: Trace the code

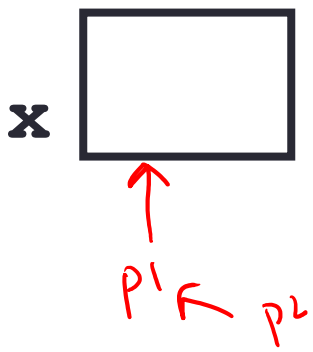
```
int x=10, y=20;
int *p1 = &x, *p2 = &y;
p2 = p1;
int **p3;
p3 = &p2;
```

# Pointer assignment

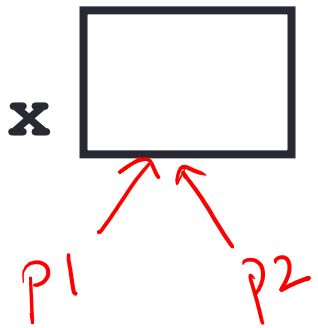
```
int *p1, *p2, x;
p1 = &x;
p2 = p1;
```

Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.




B.




C. Neither, the code is incorrect

# Swap function 2

```
int main(){
        int x= 10, y=20;
        cout<< "Before swap" <<endl;
        cout<< x<< "  " <<y<<endl;
        swap(x, y);
        cout<< "After swap" <<endl;
        cout<< x<< "  " <<y<<endl;
}
```