STRUCTS
POINTERS
REFERENCES

Problem Solving with Computers-I

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

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

How far along are you with lab04

A. Almost done
B. I am on track to finish
C. I am stuck and don’t know how to proceed
D. Haven’t started

C++ structures (lab05)

- A **`struct`** is a data structure composed of simpler data types.

```
struct Point {
    double x; //member variable of Point
    double y; //member variable of Point
};
```

- Think of Point as a new data type

```
Point p1;                 // Declare a variable of type Point
Point p1 = { 10, 20};    //Declare and initialize
```

- Access the member variables of p1 using the dot '.' operator

```
p1.x = 5;
P1.y = 10;
```

Which of the following is an/are incorrect statement(s) in C++?

```
struct Point {
    double x;
    double y;
};
```

```
struct Box {
      Point ul; // upper left corner
      double width;
      double height;
};
```

A. `ul.x = 10;`

B. `Box b1 = {{500, 800}, 10, 20};`

C. `Box b1, b2; b1.ul = {500, 500};`

D. A and C

E. None of the above are incorrect

What is printed by the code below?

```
void swapValue(int x, int y){
    int tmp = x;
    x = y;
    y = tmp;
}

int main() {

    int a=30, b=40;

     cout<<a<<" "<<b<<endl;

    swapValue( a,  b);

    cout<<a<<" "<<b<<endl;

}
```

A.

```
30 40

30 40
```

B.

```
30 40

40 30
```

C. Something else

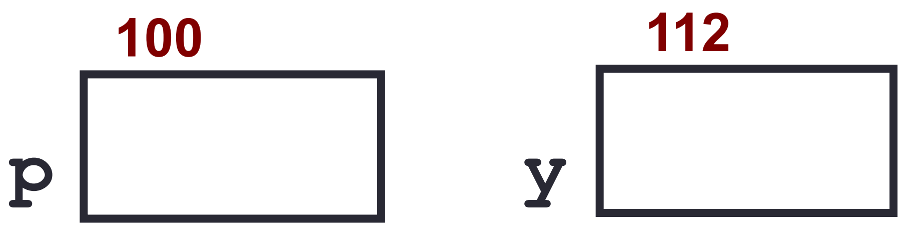
Pointers

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

```
int* p;  // Just like all uninitialized variables this will have a
         junk value

int* p = 0; //Declare and initialize
```

How to make a pointer point to something

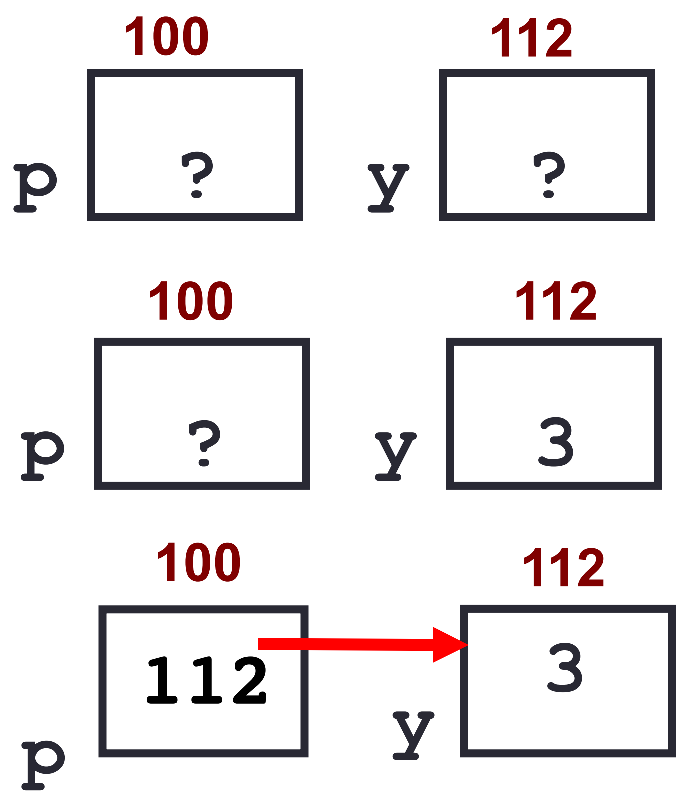
```
int *p;
int y;
```



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

How to make a pointer point to something

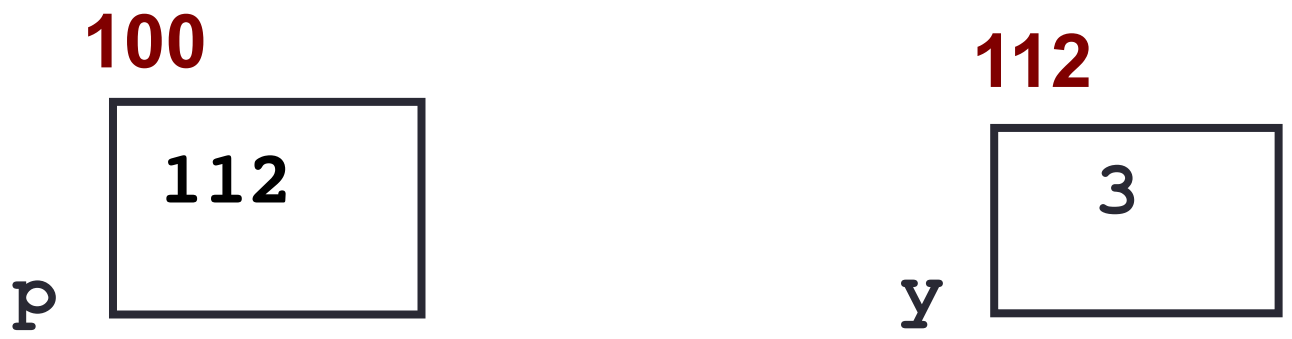
```
int *p, y;
```



`p` points to `y`

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

Pointer: p



Pointee: y

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

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

*p = 5;
```

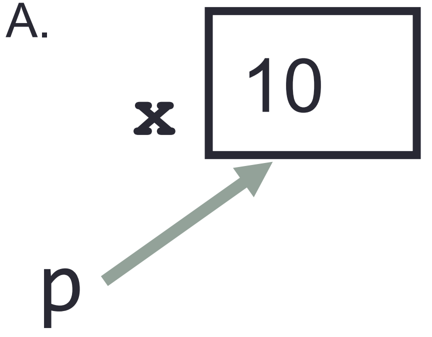
Use dereference * operator to left of pointer name

Tracing code involving pointers

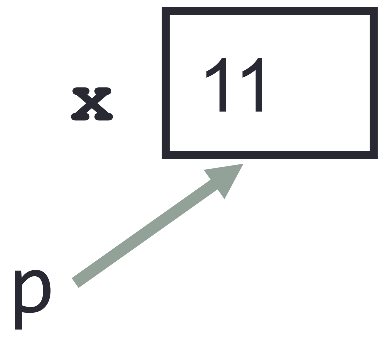
```
int *p, x=10;
p = &x;
*p = *p + 1;
```

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

A.

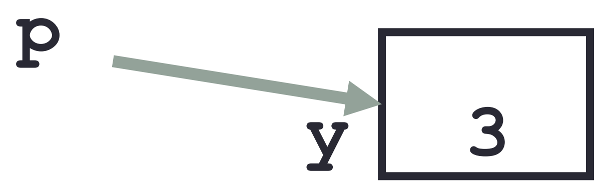


B.



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

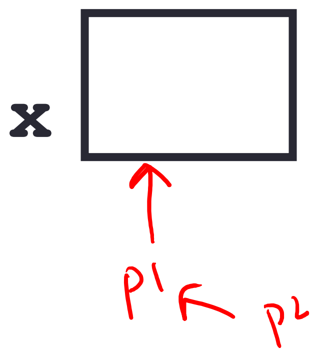
```
int x=10, y=20;

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

p2 = p1;

int **p3;

p3 = &p2;
```

Pointer assignment

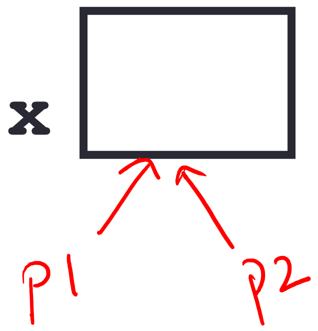
```
int *p1, *p2, x;
p1 = &x;
p2 = p1;
```

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

A.



B.

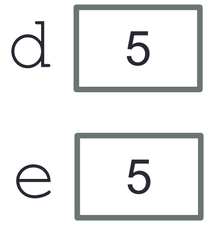


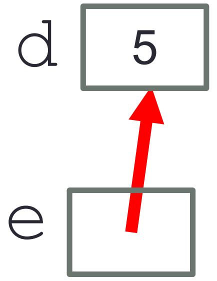
C. Neither, the code is incorrect

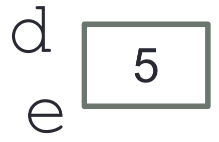
References in C++

A reference in C++ is an alias for another variable

```
int main() {
   int d = 5;
   int &e = d;
}
```

A. 

B. 

C. 

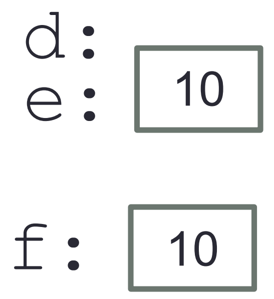
D. This code causes an error

References in C++

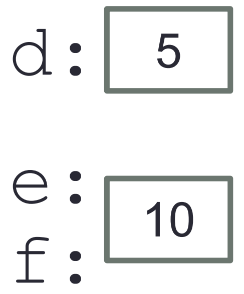
```
int main() {
   int d = 5;
   int & e = d;
   int f = 10;
   e = f;

}
```

How does the diagram change with this code?

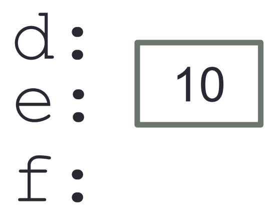
A.



B.



C.



D. Other or error

Pointers and references: Draw the diagram for this code

```
int a = 5;
int & b = a;
int* pt1 = &a;
```

What are three ways to change the value of ‘a’ to 42?

Call by reference: Modify to correctly swap a and b

```
 void swapValue(int x, int y){
     int tmp = x;
     x = y;
     y = tmp;
 }

int main() {

    int a=30, b=40;

    swapValue( a,  b);

    cout<<a<<" "<<b<<endl;

}
```

Pointers to structures

The C arrow operator (->) dereferences and extracts a structure field with a single operator.

```
struct Point {
    double x;
    double y;
};
```

Demo program using points

Next time

- Arrays and pointers
- Arrays of structs
- Dynamic memory allocation