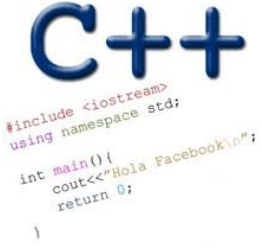
STRUCTS
PASSING STRUCTS TO FUNCTIONS

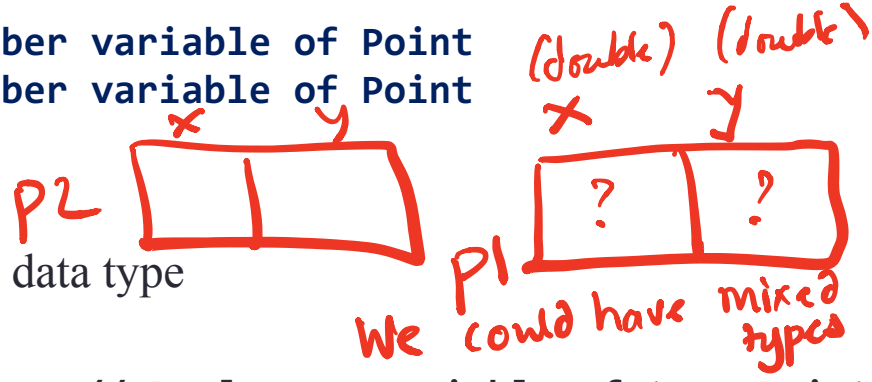
Problem Solving with Computers-I



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
```

Point p2.

void printPoint (const Point & p) {

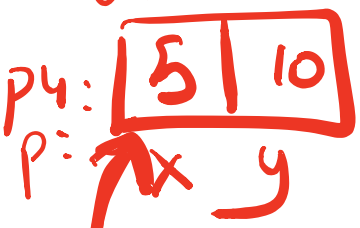
Adding the keyword const will not allow the function to change the value of p

}

void initPoint (Point *q, ...)

printPoint (p4);

//More efficient



initPoint (&p4, ...)

q

{ (*q).x = 5;
  (*q).y = 10;

shorthand

q->x = 5;
q->y = 10;

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
};
```

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

```
Point p1;
p1.x = 5;
p1.x = 10;
```

- Access via a pointer using the -> operator

```
Point* q = &p1;
(*q).x = 5;
(*q).x = 10;
q->x = 30;
```

Which of the following is/are correct 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;` // ul is a member variable of Box and can only be accessed via a Box object (b.ul.x = 10;) or pointer to a Box (p->ul.x = 10;)

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

C. Both are incorrect

D. Both statements are correct

Passing structs to functions

- Write a function that prints the x and y coordinates of a `Point`
- Write a function that takes takes two Points as input and checks if they are approximately equal

Passing structs to functions by reference

- Write a function that takes a `Point` as parameter and initializes its x and y coordinates

Arrays of structs

- Write a struct to represent a student (first name, last name, perm, major, gpa over 4 years)
- Initialize a single instance of this struct
- Write a function that takes a student as parameter and prints the following:

  Name: First last

  Major:

  Average GPA:
- Use the function to create a list of students and print their average gpa

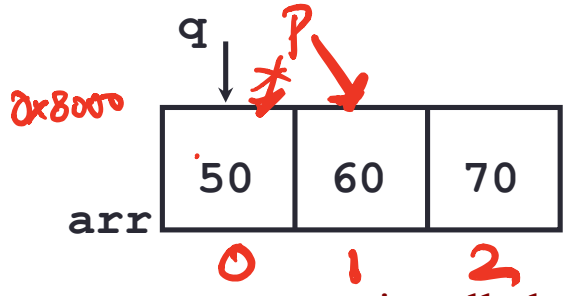
```
void IncrementPtr(int *p){
    p++;
}
```

// p = p+1; // changing p does not affect q.

main() {

```
int arr[3] = {50, 60, 70};
int *q = arr;
IncrementPtr(q);
```

}



Which of the following is true after **`IncrementPtr(q)`** is called in the above code:

A. `'q'` points to the next element in the array with value 60

B. `'q'` points to the first element in the array with value 50

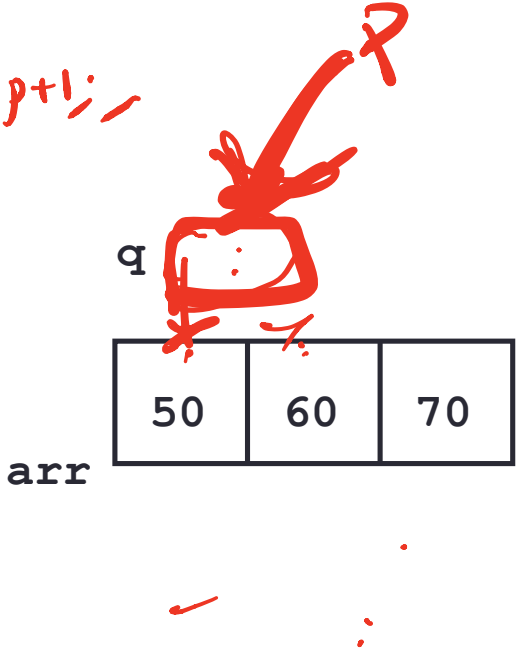
How should we implement `IncrementPtr()`, so that 'q' points to 60 when the following code executes?

```
void IncrementPtr(int **p){
    p++;
}
```

p++; Change q via p → *p = *p + 1;

```
int arr[3] = {50, 60, 70};
int *q = arr;
IncrementPtr(&q);
```

Pass q by address



A. `p = p + 1;`

B. `&p = &p + 1;`

C. `*p= *p + 1;`

D. `p= &p+1;`

Next time

- Dynamic memory allocation