

STRUCTS REFERENCES

Problem Solving with Computers-I

C++

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

int main()
cout<<"Hola Facebook!";
return 0;
}
```



References in C++

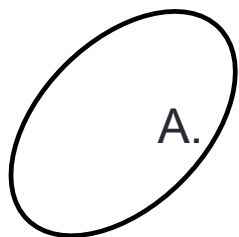
```
int main() {  
    int d = 5;  
    int &e = d;  
}
```

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

References in C++

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

How does the diagram change with this code?



d:
e:

f:

c. d:
e:
f:

B. d:

e:
f:

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;  
}
```

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;  
}
```

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
```

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.y = 10;
```

- Access via a pointer using the -> operator

```
Point* q = &p1;  
(*q).x = 5;  
(*q).y = 10;
```


Which of the following is/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, 800};`

D. A and C

E. None of the above are incorrect

Passing structs to functions

- Write a function that prints the x and y coordinates of a `Point`

Passing structs to functions by reference

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

Pointer Arithmetic Question

How many of the following are invalid?

- I. pointer + integer ($\text{ptr}+1$)
- II. integer + pointer ($1+\text{ptr}$)
- III. pointer + pointer ($\text{ptr} + \text{ptr}$)
- IV. pointer – integer ($\text{ptr} - 1$)
- V. integer – pointer ($1 - \text{ptr}$)
- VI. pointer – pointer ($\text{ptr} - \text{ptr}$)
- VII. compare pointer to pointer ($\text{ptr} == \text{ptr}$)
- VIII. compare pointer to integer ($1 == \text{ptr}$)
- IX. compare pointer to 0 ($\text{ptr} == 0$)
- X. compare pointer to NULL ($\text{ptr} == \text{NULL}$)

#invalid

A: 1

B: 2

C: 3

D: 4

E: 5

Pointer Arithmetic Question

How many of the following are invalid?

- I. pointer + integer ($ptr+1$)
- II. integer + pointer ($1+ptr$)
- ~~III. pointer + pointer ($ptr + ptr$)~~
- IV. pointer - integer ($ptr - 1$)
- ~~V. integer - pointer ($1 - ptr$)~~
- VI. pointer - pointer ($ptr - ptr$)
- VII. compare pointer to pointer ($ptr == ptr$)
- ~~VIII. compare pointer to integer ($1 == ptr$)~~
- IX. compare pointer to 0 ($ptr == 0$)
- X. compare pointer to NULL ($ptr == NULL$)

#invalid

A: 1

B: 2

C: 3

D: 4

E: 5