Data Structures and Algorithms
(CS221)

Pointers Arithmetic

Pointer and Array

An array name acts like a pointer to its first element

Use pointers to access and manipulate array elements

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

This means **arr** holds the memory address of **arr[0]**

The array name **arr** is a constant pointer to the first element (**arr[0]**).

Pointer and Array

```
int myarray [20];
int * mypointer;
```

Which statement is valid?

```
mypointer =myarray;

myarray =mypointer;
```

Pointer and Array (Example)

```
1       int main () {
2       int numbers[5];
3       int * p;
4       p = numbers;
5       *p = 10;
6       p++;
7       *p = 20;
8       p = &numbers[2];
9       *p = 30;
10      p = numbers + 3;
11      *p = 40;
12      p = numbers;
13      *(p+4) = 50;

14      for (int n=0; n<5; n++)
15         cout << numbers[n] << ", ";
16      return 0;
17  }
```

Pointer and Array (Example)

```
int   b[10]={0,10,20,30,40,50,60,70,80,90};
int   *ptr=&b[0];          cout<< ptr<<"\n";
                           cout<< *ptr<<"\n";
ptr=b+2 ;                  cout<<  ptr<<"\n";
                           cout<< *ptr<<"\n";
ptr=ptr+2 ;                cout<<  ptr<<"\n";
                           cout<< *ptr<<"\n";
ptr=&b[6];                 cout<< ptr<<"\n";
                           cout<< *ptr<<"\n";
ptr++;                     cout<< ptr<<"\n";
                           cout<< *ptr<<"\n";
                           cout<< *b+1<<"\n";
                           cout<< *(b+1)<<"\n";
                           cout<< (b+1)<<"\n";
                           cout<< *(ptr+1)<<"\n";
                           cout<< (ptr+1)<<"\n";
                           cout<< b[3]<<"\n";
                           cout<< &b[3]<<"\n";
          getch();
```

```
----------→ 0x0000A000
----------→ 0
----------→ 0x0000A008
----------→ 20
----------→ 0x0000A010
----------→ 40
----------→ 0x0000A018
----------→ 60
----------→ 0x0000A01C
----------→ 70
----------→ 1
----------→ 10
----------→ 0x0000A004
----------→ 80
----------→ 0x0000A020
----------→ 30
----------→ 0x0000A00C
```

Pointer and String

```
char str[] = ”World";
```

Automatically includes a null character (\0) at the end.

strings can be manipulated using **character arrays** or **pointers to characters**.

The name str is a **pointer** to the first character.

```
char *ptr = str;  // ptr now points to   ‘W

cout << *(ptr + 1);    // Outputs   ‘o'
```

Pointer and String (example)

```
char str[30]; strcpy(str,"Ex: Ptrs. & Strings.");
char   *ptr=str;  cout<<str<<"\n";
                  cout<< &str<<"\n";
                  cout<< *str<<"\n";
                  cout<< ptr<<"\n";
                  cout<< *ptr<<"\n";
       ptr++;     cout<< ptr<<"\n";
                  cout<< *ptr<<"\n";
   ptr=str+3;     cout<< ptr<<"\n";
                  cout<< *ptr<<"\n";
    ptr=ptr+5;    cout<< ptr<<"\n";
                  cout<< *ptr<<"\n";
```

----→ Ex: Ptrs. & Strings.

----→ 0x00001000

----→ E

----→ Ex: Ptrs. & Strings.

----→ E

----→ x: Ptrs. & Strings.

----→ x

----→ Ptrs. & Strings.

----→ EMPTY_SPACE

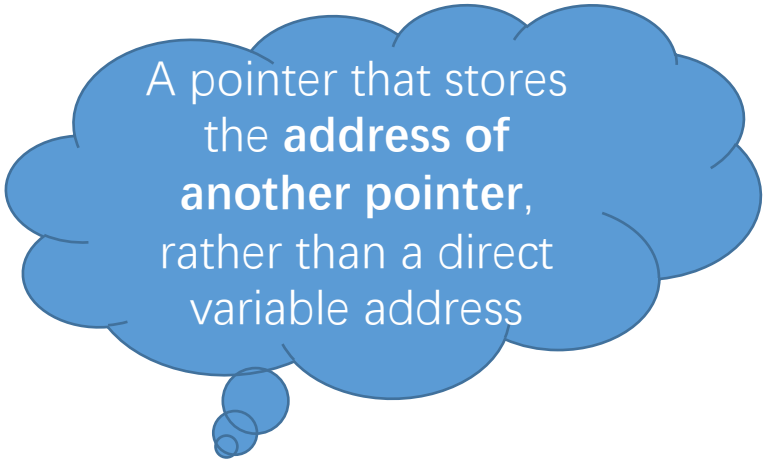
----→ . & Strings..

----→ .

Pointer and String (example)

```
char str[30]; strcpy(str,"Ex: Ptrs. & Strings.");
 char   *ptr=str;
          ptr=&str[12];        cout<< ptr<<"\n";          ----------→ Strings
                               cout<< *ptr<<"\n ";        ----------→ S
                               cout<< *str+2<<"\n";       ----------→ 71
                               cout<< char(*str+2)<<"\n  "; -----→ G
                               cout<< *(str+2)<<" \n";    ----------→ :
                               cout<< (str+2)<<" \n";     ----------→ : Ptrs. & Strings.
                               cout<< *(ptr+1)<<" \n";    ----------→ t
                               cout<< (ptr+1)<<" \n";     ----------→ trings .
                               cout<< str[4]<<" \n";      ----------→ P
                               cout<< &str[4]<<" \n";     ----------→ Ptrs. & Strings
                              cout<< &str[1]<<" \n";      ----------→ Ex: Ptrs. & Strings.
                               getch();
```

Pointer to Pointer (Double Pointer)

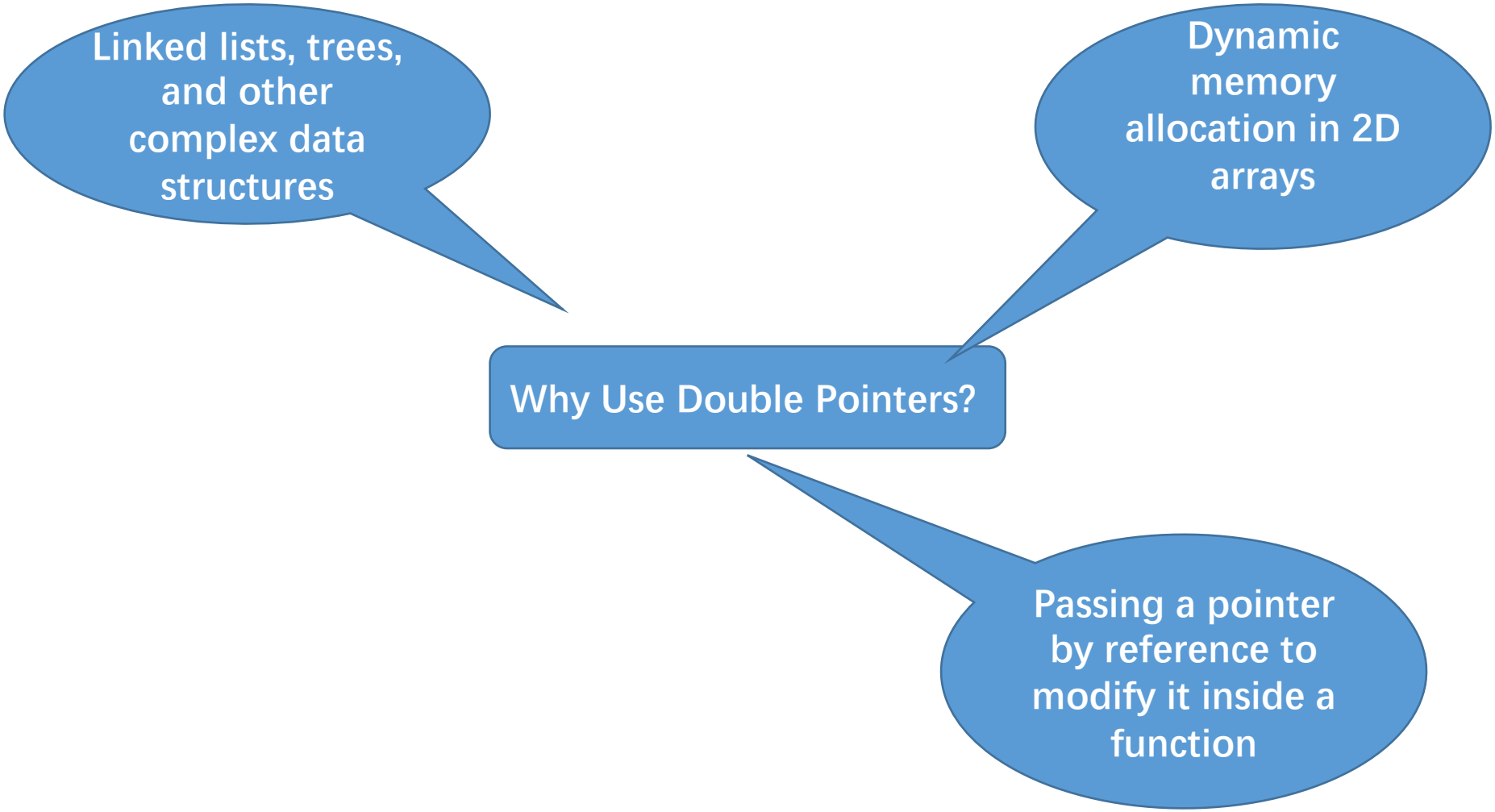


In simple terms:

- A **single pointer (*p)** stores the address of a variable.
- A **double pointer (**pp)** stores the address of a single pointer.

```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```

Pointer to Pointer (Double Pointer)



Pointer to Pointer (Double Pointer)

```
char a;          char *b;          char **c;
a = 'z   ';          b = &a;           c = &b;
                  cout << &a;
                  cout << &b;
                  cout << &c;
                  cout << a;
                  cout << b;
                  cout << c;
                  cout << *a;
                  cout << *b;
                  cout << *c;
                  cout << **c;
        getch();
}
```

------→ 0x0000B001
------→ 0x0000C001
------→ 0x0000D001
------→ z
------→ 0x0000B001
------→ 0x0000C001
------→ Error
------→ z
------→ 0x0000B001
------→ z

Pointers and Structures

```
struct student{
        int id;
        char name[20];
        };
int main(){
        student st, *st_ptr = &st;
        st.id = 10;
        strcpy(st.name, "Faran Khalid");
        cout<<"\n";
        cout<<"\nStudent Id:  "<<st_ptr->id;
        cout<<"\nStudent Name:  "<<st_ptr->name;
        getch(); return 0;
        }
```

Questions?

zahmaad.github.io