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)

```
int main () {
    int numbers[5];
    int * p;
    p = numbers;
    *p = 10;
    p++;
    *p = 20;
    p = &numbers[2];
    *p = 30;
    p = numbers + 3;
    *p = 40;
    p = numbers;
    *(p+4) = 50;

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

Pointer and Array (Example)

```
int   b[10]={0,10,20,30,40,50,60,70,80,90};
int   *ptr=&b[0];         cout<< ptr<<"\n";           ----------→  0x0000A000
                           cout<< *ptr<<"\n";          ----------→  0
ptr=b+2 ;                 cout<<  ptr<<"\n";          ----------→  0x0000A008
                          cout<< *ptr<<"\n";          ----------→  20
ptr=ptr+2 ;               cout<<  ptr<<"\n";          ----------→  0x0000A010
                          cout<< *ptr<<"\n";          ----------→  40
ptr=&b[6];                cout<< ptr<<"\n";           ----------→  0x0000A018
                          cout<< *ptr<<"\n";          ----------→  60
ptr++;                     cout<< ptr<<"\n";           ----------→  0x0000A01C
                           cout<< *ptr<<"\n";          ----------→  70
                            cout<< *b+1<<"\n";         ----------→  1
                             cout<< *(b+1)<<"\n";      ----------→  10
                             cout<< (b+1)<<"\n";       ----------→  0x0000A004
                            cout<< *(ptr+1)<<"\n";     ----------→  80
                             cout<< (ptr+1)<<"\n";     ----------→  0x0000A020
                             cout<< b[3]<<"\n";        ----------→  30
                             cout<< &b[3]<<"\n";       ----------→  0x0000A00C
                getch();
```

Pointer and String

```
char str[] = ”World";
```

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

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

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.");   ----→   Ex: Ptrs. &
char   *ptr=str;  cout<<str<<"\n";                         Strings.
                       cout<< &str<<"\n";   ----→   0x00001000
                       cout<< *str<<"\n";   ----→   E
                       cout<< ptr<<"\n";    ----→   Ex: Ptrs. & Strings.
                       cout<< *ptr<<"\n";   ----→   E
          ptr++;       cout<< ptr<<"\n";    ----→   x: Ptrs. & Strings.
                       cout<< *ptr<<"\n";   ----→   x
      ptr=str+3;     cout<< ptr<<"\n";      ----→    Ptrs. & Strings.
                       cout<< *ptr<<"\n";   ----→   EMPTY_SPACE
     ptr=ptr+5;      cout<< ptr<<"\n";      ----→   .  & Strings..
                       cout<< *ptr<<"\n";   ----→   .
```

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)

A pointer that stores the **address of another pointer**, rather than a direct variable address

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)

Why Use Double Pointers?

Linked lists, trees, and other complex data structures

Dynamic memory allocation in 2D arrays

Passing a pointer by reference to modify it inside a function

Pointer to Pointer (Double Pointer)

```
char a;          char *b;         char **c;
a = 'z   ';         b = &a;          c = &b;
                 cout << &a;      ------→     0x0000B001
                 cout << &b;      ------→     0x0000C001
                 cout << &c;      ------→     0x0000D001
                 cout << a;       ------→       z
                 cout << b;       ------→     0x0000B001
                 cout << c;       ------→     0x0000C001
                 cout << *a;      ------→     Error
                 cout << *b;      ------→     z
                 cout << *c;      ------→     0x0000B001
                 cout << **c;     ------→     z
      getch();
}
```

Passing Pointers to Functions

```
void F1(float a, float b){
     a++;  b++;
     }
void F2(float &a, float &b){
     a++;  b++;
}
void F3(float *a, float *b){
     *a=*a-1;  *b=*b-1;     }
```

```
void F1(int a, int b){
    int temp;
    temp=a; a=b; b=temp;
}
void F2(int &a, int &b){
       int tempA =a;
       a=tempA / b;      b=tempA % b;
}
void F3(int *a, int*b){
    int *tempA=a;
    *a = *tempA  +  *b;    *b = *tempA  -  *b;
 }
```

```
void main(){
    int x=9, y=5;    float a=9,b=5
    int *ptrX=&x;     int *ptrY=&y;
    float *ptrA=&a;     float *ptrB=&b;

    F1(a,b);                    cout<<"\na: "<<a<<"\tb: "<<b;
    F2(a,b);                    cout<<"\na: "<<a<<"\tb: "<<b;
    F3(ptrA,ptrB);              cout<<"\na: "<<a<<"\tb: "<<b;
    F3(&a,&b);                  cout<<"\na: "<<a<<"\tb: "<<b;

     F1(x,y);                   cout<<"\nx: "<<x<<"\ty: "<<y;
    F2(x,y);                    cout<<"\nx: "<<x<<"\ty: "<<y;
    F3(ptrX,ptrY);              cout<<"\nx: "<<x<<"\ty: "<<y;
    F3(&x,&y);                  cout<<"\nx: "<<x<<"\ty: "<<y;
```

```
a:  9        b:  5
a:  10       b:  6
a:  9        b:  5
a:  8        b:  4

 x:  9       y:  5
 x:  1       y:  4
 x:  5       y:  1
 x:  6       y:  5
```

Passing Pointers to Functions

```
void F1(float a, float b){
      a++;  b++;
      }
void F2(float &a, float &b){
      a++;  b++;
}
void F3(float *a, float *b){
      *a=*a-1;  *b=*b-1;     }
```

```
void F1(int a, int b){
     int temp;
     temp=a; a=b; b=temp;
}
void F2(int &a, int &b){
        int tempA =a;
        a=tempA / b;      b=tempA % b;
}
void F3(int *a, int*b){
     int *tempA=a;
     *a = *tempA  +  *b;    *b = *tempA  -  *b;
 }
```

```
void main(){
     int x=9, y=5;    float a=9,b=5
   int *ptrX=&x;     int *ptrY=&y;
     float *ptrA=&a;     float *ptrB=&b;
     x = 9;  y = 5; a = 13; b = 6;
    F1(*ptrX,*ptrY);          cout<<"\nx: "<<x<<"\ty: "<<y;
    F1(*ptrA,*ptrA);          cout<<"\na: "<<a<<"\tb: "<<b;
    F2(*ptrX,*ptrY);          cout<<"\nx: "<<x<<"\ty: "<<y;
    F2(*ptrA,*ptrA);          cout<<"\na: "<<a<<"\tb: "<<b;

    F1(ptrX,ptrY);            cout<<"\nx: "<<x<<"\ty: "<<y;
    F1(ptrA,ptrA);            cout<<"\na: "<<a<<"\tb: "<<b;
    F2(ptrX,ptrY);            cout<<"\nx: "<<x<<"\ty: "<<y;
    F2(ptrA,ptrA);            cout<<"\na: "<<a<<"\tb: "<<b;
    F3(*ptrX,*ptrY);          cout<<"\nx: "<<x<<"\ty: "<<y;
    F3(*ptrA,*ptrA);          cout<<"\na: "<<a<<"\tb: "<<b;
```

```
x:  9        y:  5
a:  13       b:  6
x:  1        y:  4
a:  15       b:  6

Error
Error
Error
Error
Error
Error
```

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