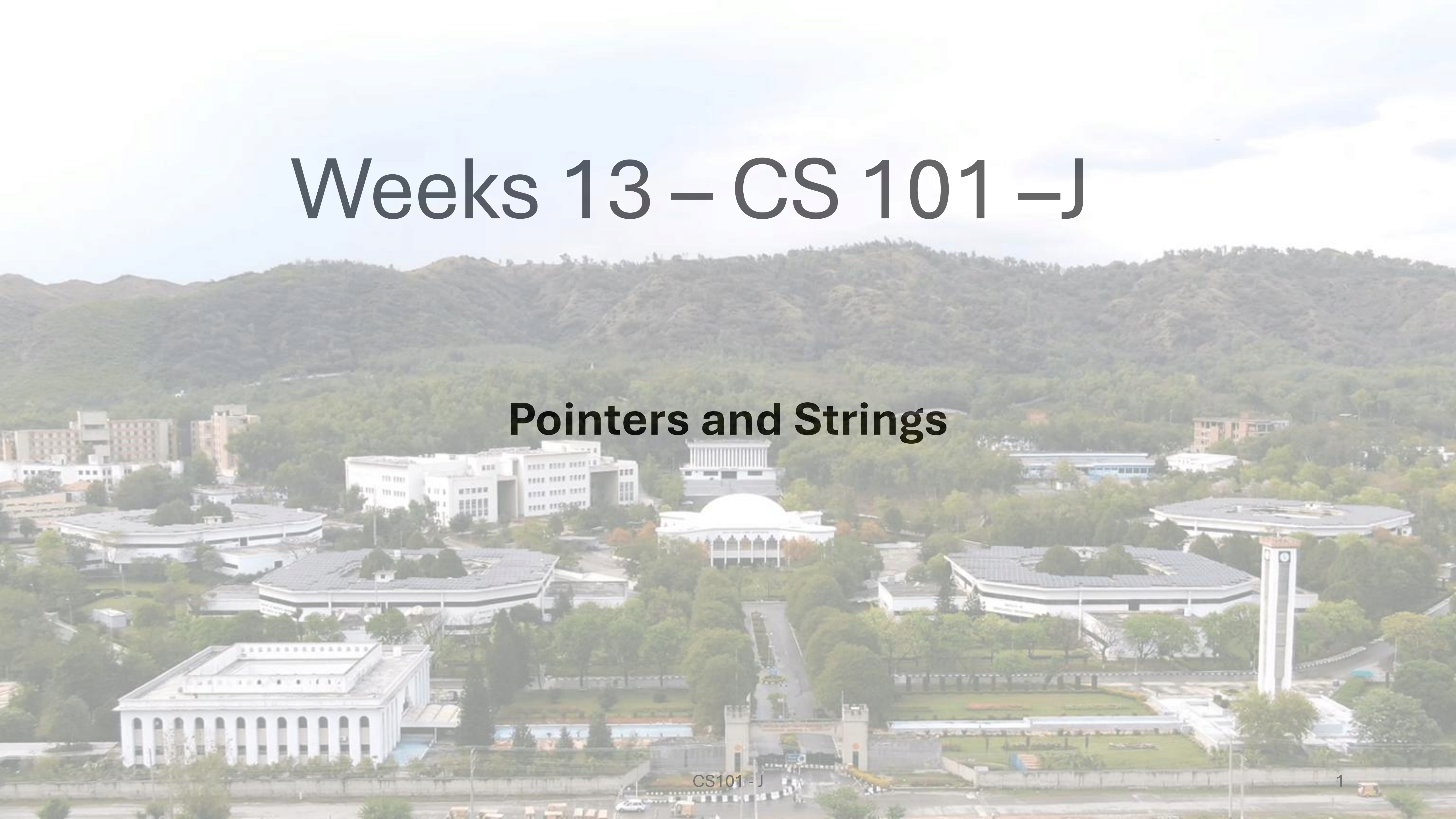


Weeks 13 – CS 101 –J

Pointers and Strings

Example 1 – Simple Example of Pointer

```
int main()
{
  int x = 10;
  int* y = &x;
    cout << *y;
    return 0;
}
```

Prints the value 10 – the value in the variable at the address stored in **y**

In other words, the value to which **y** points

Example 2 – Change value of variable

```
int main()
{
  int x = 10;
  int* y = &x;
  x = 20;
  cout << *y;
    return 0;
}
```

Prints the value 20

Example 3 – Change value using pointer

```
int main()
{
  int x = 10;
  int* y = &x;
  *y = 20;
  cout<<x<<*y;
   return 0;
}
```

Prints: 20, 20 – we can use *y as a left-hand value, which changes the contents of the address that **y** points to

Example 4 – Compiler Error

```
int main()
{
  int x = 10;
  int* y = &x;
  y = 20;
  cout<<x<<*y;
  return 0;
}
```

Prints: 10 ??? – the second term will be what ever happens to be at bytes 20, 21, 22 and 23 – could be junk.

But using Dev C++ , the code will not compile

Example 5 – Perfect Code (but not correct values)

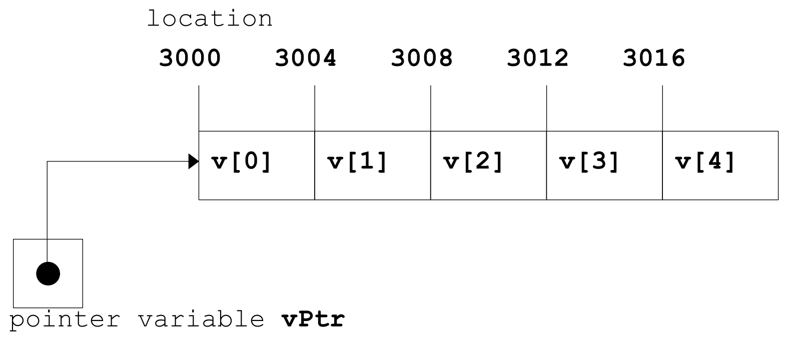
```
int main()
{
  int x = 10;
  int* y = &x;
  y++;
  cout<<x<<*y;
  return 0;
}
```

Prints: 10 ??? – the second term will be what ever happens to be at the next bytes - could be junk.

In Dev C++ , the code will compile (Yes)

Pointer Expressions and Pointer Arithmetic

- Pointer arithmetic
  - Increment/decrement pointer (`++` or `--`)
  - Add/subtract an integer to/from a pointer(`+` or `+=` , `-` or `-=`)
  - Pointers may be subtracted from each other
  - Pointer arithmetic is meaningless unless performed on an array
- 5 element **`int`** array on a machine using 4 byte **`int`s**
  - **`vPtr`** points to first element **`v[ 0 ]`**, which is at location 3000
    - **`vPtr = 3000`**
  - **`vPtr += 2`**; sets **`vPtr`** to **`3008`**
    - **`vPtr`** points to **`v[ 2 ]`**



Pointer Expressions and Pointer Arithmetic

- Subtracting pointers
  - Returns the number of elements between two addresses

```
vPtr2 = v[ 2 ];
vPtr = v[ 0 ];
vPtr2 - vPtr == 2
```

- Pointer comparison
  - Test which pointer points to the higher numbered array element
  - Test if a pointer points to **0** (**`NULL`**)

```
if ( vPtr == ‘0’ )
   statement
```

Example 1

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

int main()
{
		char string1[]="GIKI";
		char *aptr;
		int i;
		aptr=&string1[0];

		for (i=0; i<4; i++)
				cout<<"The value of element wise a is "<<aptr[i]<<endl;

		cout<<"After the loop printing the array in 1 iteration"<<endl;
		cout<<"The value of a is "<<aptr<<endl;

		return 0;
}
```

The Relationship Between Pointers and Arrays

- Arrays and pointers closely related
    - Array name like constant pointer
    - Pointers can do array subscripting operations
    - Having declared an array `b[ 5 ]` and a pointer `bPtr`
        - `bPtr` is equal to `b`

          ```
          bptr == b
          ```
        - `bptr` is equal to the address of the first element of `b`

          ```
          bptr == &b[ 0 ]
          ```

The Relationship Between Pointers and Arrays

- Accessing array elements with pointers
  - Element `b[ n ]` can be accessed by `*( bPtr + n )`
    - Called pointer/offset notation
  - Array itself can use pointer arithmetic.
    - `b[ 3 ]` same as `*(b + 3)`
  - Pointers can be subscripted (pointer/subscript notation)
    - `bPtr[ 3 ]` same as `b[ 3 ]`

Example 2

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

int main()
{
        char string1[]="GIKI";
        char *aptr;
        int i;
        aptr=&string1[0];

        for (i=0; i<4; i++)
                cout<<"The value of element wise a is "<<*(aptr+i)<<endl;

        cout<<"After the loop printing the array in 1 iteration"<<endl;
        cout<<"The value of a is "<<aptr<<endl;

        return 0;
}
```

- Execute wk13s13.cpp and understand it

```
#include <iostream>
using namespace std;
int main()
{
        int arr[]={1,2,3,4,5};
        int x=10;
        int *ptr;

        // pointer pointing to x
        ptr=&x; // Remember: ptr=x is compiler error
        cout<<x<<" "<<*ptr<<" "<<ptr<<endl;

        // pointer pointing to array
        ptr=arr;//          ptr=&arr[0] is ok
        //ptr=arr[0]; //is wrong

        cout<<ptr[0]<<" "<<ptr[1]<<" "<<ptr[4]<<endl;

        return 0;
}
```

- Understand wk13s14.cpp
- Difference between char pointer and other pointers

```
#include <iostream>
using namespace std;
int main()
{
        int i_array[]={1,2,3,4,5};
        char c_array[]="FCSE";

        int *iptr;
        char *cptr;

        iptr=i_array;
        cptr=c_array;

        cout<<cptr<<endl;
        cout<<iptr<<endl;

        return 0;
}
```

Summary

- int pointer can point to int variable or int array
  - int *ptr; int x; ptr=&x;
  - int *ptr; int x[4]; ptr=x;

- char pointer can point to char variable or char array

- But one special point about char pointer
  - cout<<charptr -- this displays the whole character array instead of starting address

Example 3

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

int main()
{
        char *sptr[4]={"GIKI","FME","FCSE","Guest House"};
        int i;
        for(i=0;i<4;i++)
                cout<<"The value is "<<*(sptr+i)<<endl;
        return 0;
}
```