



Pointer Arithmetic

Chapter 12



Objectives

- You will be able to
 - use a pointer to access any element of an array.
 - use pointer arithmetic appropriately.



Pointers and Arrays

- Remember that the name of an array represents the address of the first entry.
 - Same as a pointer.
- A pointer can be set to the address of an array.

```
int values[20];  
int *pValues;  
...  
pValues = values;
```



Pointers and Arrays

- Pointers and array names can be used interchangeably.
- When a pointer holds the base address of an array, we can put an index after it to refer to any element of the array.

Note no
“*”

```
pValues[10] = 201;
```

- Same effect as

```
values[10] = 201;
```



Using a pointer with an index

```
#include <stdio.h>
```

```
int main ( )
```

```
{
```

```
    int some_numbers[] = {101, 102, 103, 104, 105};
```

```
    int length = sizeof(some_numbers) / sizeof(some_numbers[0]);
```

```
    int i;
```

```
    int* pNumber = some_numbers;
```

```
    for (i = 0; i < length; i++)
```

```
    {
```

```
        printf ("Entry %d contains %d\n", i, pNumber[i] );
```

```
    }
```

```
    return 0;
```

```
}
```

Note no
"*"

Same as
some_numbers[i]



Using a pointer with an index

```
turnerr@login0:~/test
[turnerr@login0 test]$
[turnerr@login0 test]$
[turnerr@login0 test]$ gcc -Wall test.c
[turnerr@login0 test]$ ./a.out
Entry 0 contains 101
Entry 1 contains 102
Entry 2 contains 103
Entry 3 contains 104
Entry 4 contains 105
[turnerr@login0 test]$
```



Pointer Arithmetic

- We can also increment and decrement a pointer.

```
int values[20];
```

```
int* pValues;
```

```
...
```

```
pValues = values;    pValues points to  
                      values[0]
```

```
pValues += 1;        pValues points to  
                      values[1]
```

- The compiler knows the size of whatever the pointer points to and increments the address in the pointer appropriately.



Pointer Arithmetic

- Incrementing and decrementing a pointer only makes sense when the pointer points to an array.
- Increment says move forward that many entries.
- Decrement says move back that many entries.
- There is no check that the result is a valid reference to the array!



Why Do Pointer Arithmetic?

- May be slightly more efficient.
 - Fewer instructions executed than stepping through an array with an integer index.
 - Normally not significant.
- Slightly more compact notation.
- Closer to programming in machine language.
 - Traditional C culture.

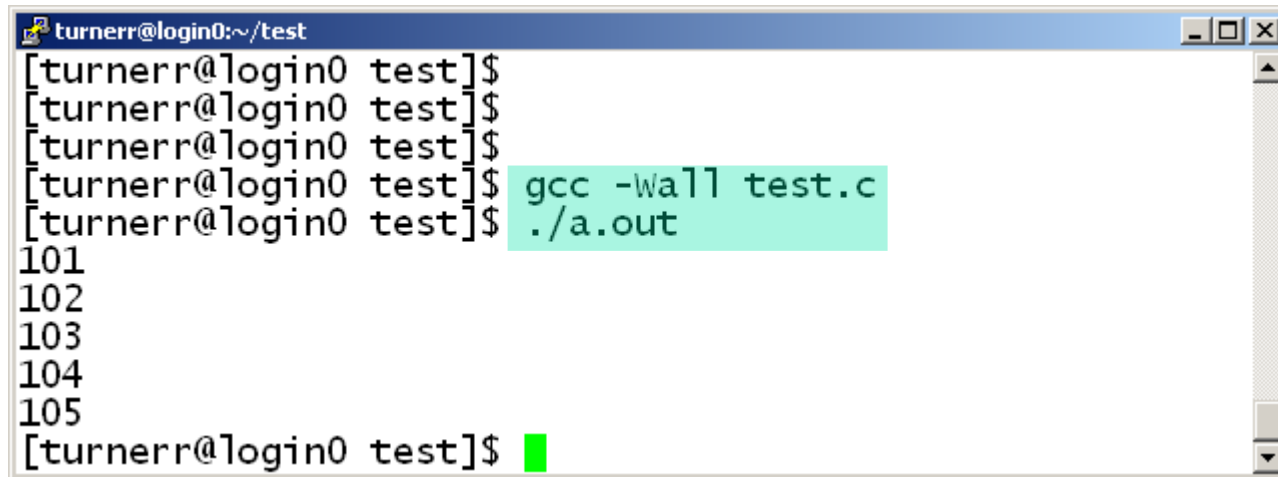


Pointer Arithmetic Example

```
int main ( )
{
    int some_numbers[] = {101, 102, 103, 104, 105};
    int* pNumber = some_numbers; Initialize pointer to address of first element of
                                   array
    while (pNumber <= &some_numbers[4]) Continue while pointer inside
    {                                     array
        printf ("%d\n", *pNumber ); Contents of the address that pNumber
        pNumber++; Advance pNumber to next element of the points to
    }               array

    return 0;
}
```

Compile and Run



```
turnerr@login0:~/test
[turnerr@login0 test]$
[turnerr@login0 test]$
[turnerr@login0 test]$
[turnerr@login0 test]$ gcc -Wall test.c
[turnerr@login0 test]$ ./a.out
101
102
103
104
105
[turnerr@login0 test]$
```

Same
result.



Invalid Array References

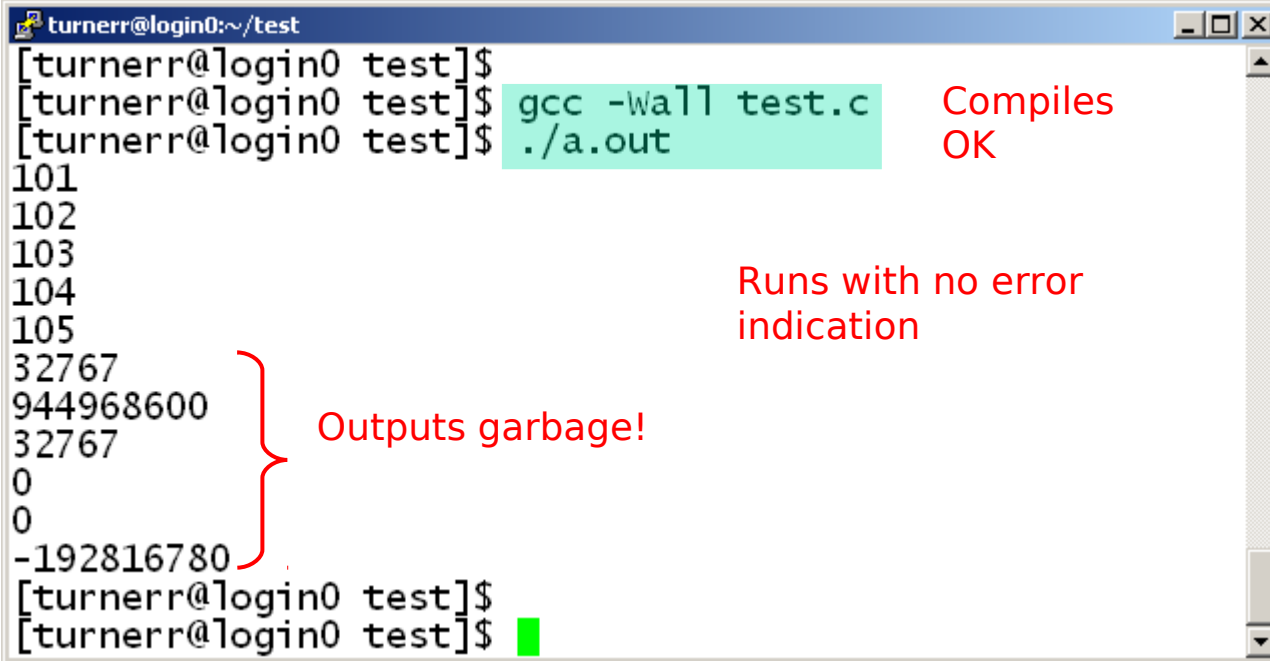
```
int main ( )
{
    int some_numbers[] = {101, 102, 103, 104, 105};
    int* pNumber = some_numbers;

    while (pNumber <= &some_numbers[10])
    {
        printf ("%d\n", *pNumber );
        pNumber++;
    }

    return 0;
}
```

Go on incrementing
pNumber past end of the
array.

Compile and Run



```
turnerr@login0:~/test
[turnerr@login0 test]$
[turnerr@login0 test]$ gcc -Wall test.c
[turnerr@login0 test]$ ./a.out
101
102
103
104
105
32767
944968600
32767
0
0
-192816780
[turnerr@login0 test]$
[turnerr@login0 test]$
```

Compiles
OK

Runs with no error
indication

Outputs garbage!



Pointer Arithmetic

- Pointers can be increment only by *integer* values.
- For example, it doesn't make sense to add two pointers.



Attempt to add two pointers

```
main ( )
{
    int some_numbers[] = {101, 102, 103, 104, 105};
    int length = sizeof(some_numbers) /
sizeof(some_numbers[0]);
    int i;
    int* pNumber = some_numbers;
    int* pNumber2 = &some_numbers[2];

    for (i = 0; i <= 10; i++)
    {
        printf ("Entry %d contains %d\n", i, *pNumber);
        pNumber += pNumber2;
    }
    return 0;
}
```

This gets a compile
error



Attempt to add two pointers

```
turnerr@login0:~/test
[turnerr@login0 test]$
[turnerr@login0 test]$
[turnerr@login0 test]$
[turnerr@login0 test]$ gcc -Wall test.c
test.c: In function 'main':
test.c:12: error: invalid operands to binary +
[turnerr@login0 test]$
[turnerr@login0 test]$
[turnerr@login0 test]$
```




Pointer Arithmetic

- We can *subtract* one pointer from another.

- Only makes sense when both point into the same array.

```
int values[20];  
int* pValues1 = &values[1];  
int* pValues2 = &values[2];  
int diff;  
...  
diff = pValues2 - pValues1;
```

- Sets diff to 1
 - Difference in units of array entries



Subtracting Pointers

```
#include <stdio.h>
```

```
int main ( )
```

```
{
```

```
    int some_numbers[] = {101, 102, 103, 104, 105};
```

```
    int* pNumber = some_numbers;
```

```
    int* pNumber2 = &some_numbers[2];
```

```
    int diff = pNumber2 - pNumber;
```

```
    printf ("Difference is %d\n", diff);
```

```
    return 0;
```

```
}
```



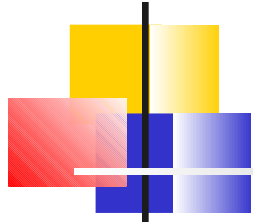
Subtracting Pointers

```
turnerr@login0:~/test
[turnerr@login0 test]$
[turnerr@login0 test]$ gcc -Wall test.c
[turnerr@login0 test]$ ./a.out
Difference is 2
[turnerr@login0 test]$
[turnerr@login0 test]$
```



Summary

- When a pointer holds the address of an array, or an element of an array:
 - We can use the pointer as if it were an array name.
 - `pMyArray[i]`
 - We can increment and decrement the pointer.
 - `pMyArray++`
- When two pointers hold addresses of elements in the same array, we can subtract one from the other.
 - `diff = pMyArray1 - pMyArray2;`



Pointer Arithmetic

- Pointer arithmetic is a rich source of errors in C programs.

(My Opinion)

- It should generally be avoided.
 - Better to use array index values when possible.
 - Do normal integer arithmetic on index values.
 - May use more CPU time than using pointer arithmetic, but less error prone.
- But it is widely used by C programmers.
 - You need to understand it!