CSC180: Lecture 19

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Pointer Pitfall 3: incorrect pointer comparison

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

int main ()
{
 int *p = NULL, *q = NULL;
 p = (int *) malloc (sizeof (int));
 q = (int *) malloc (sizeof (int));
 if ( p < q )
  printf ("\n p is less than q");     // what are you trying to do??
 else
  printf ("\n p is greater than q");
 return 0;
}
```

Pointer Pitfall 4: failure to do pointer re-initialization

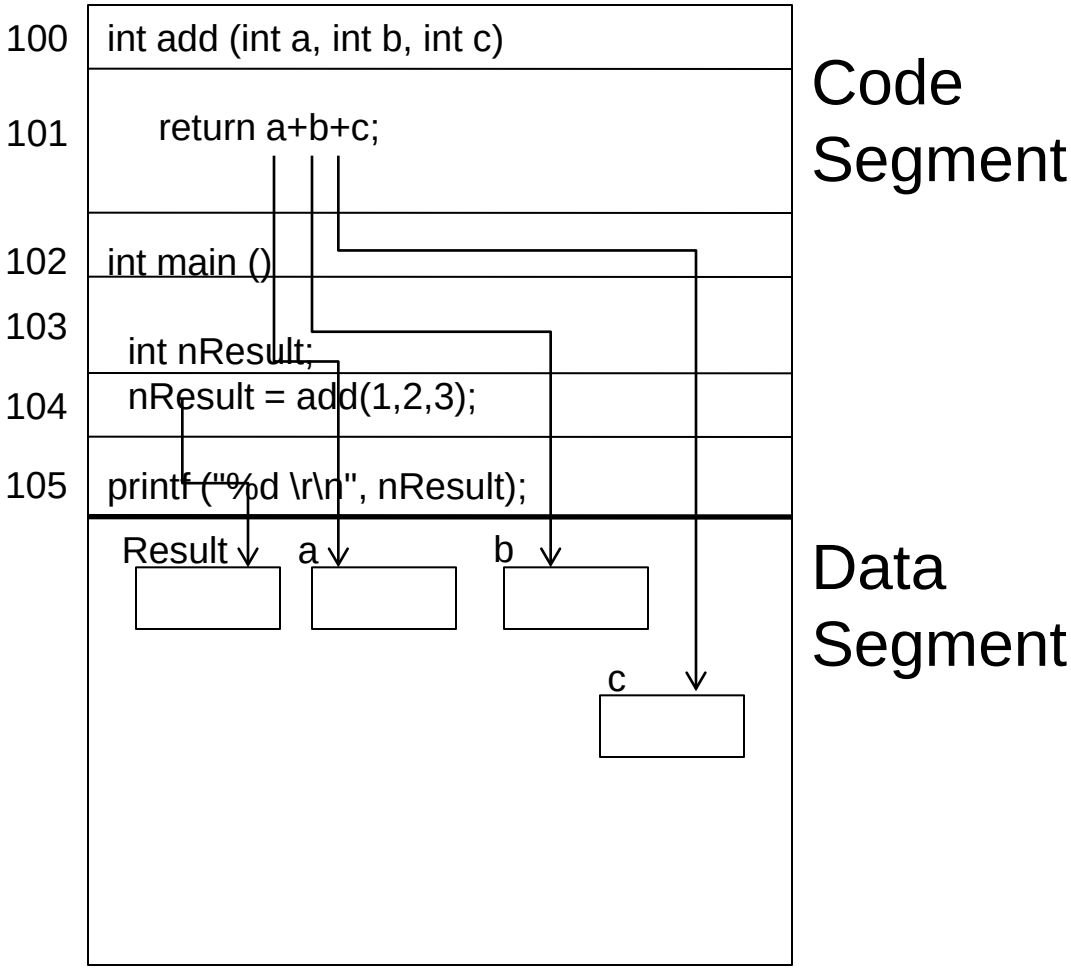
```
#include <stdio.h>
#include <stdlib.h>

int main () {
 int i;
 int *p = (int *) malloc ( sizeof (int) * 10 );




 for ( i = 0; i < 10; i = i +1)  {
      *p = i;
       p = p + 1;
   }

for ( i = 0; i < 10; i = i +1)  {
      printf (" %d ", *p); //Prints Garbage Value
      p = p + 1;
   }

  return 0;
}
```

Pointer Pitfall 4: correction

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

int main ()  {
 int i;
 int *p = (int *) malloc ( sizeof (int) * 10 );
 int *q;

 q = p;
 for ( i = 0; i < 10; i = i +1)  {
      *p = i;
       p = p + 1;
  }

 p = q;
 for ( i = 0; i < 10; i = i +1)  {
      printf (" %d ", *p); //Prints Value 1 to 10;
      p = p + 1;
   }

  return 0;
}
```

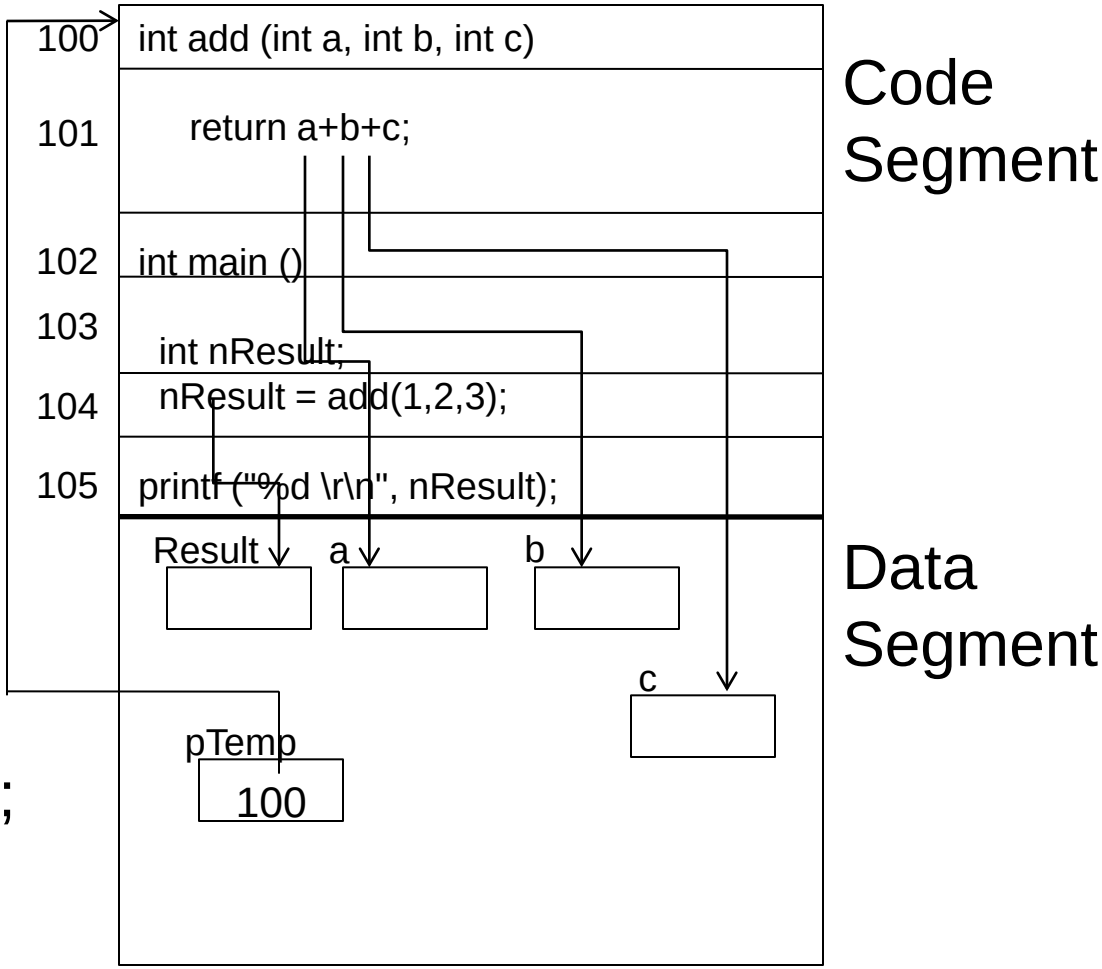
Memory Organization

```
int add (int a, int b, int c)
{
    return a+b+c;
}

int main ()
{
  int nResult;
  nResult = add(1,2,3);
  printf ("%d \r\n", nResult);
}
```



Memory Organization

```
int add (int a, int b, int c)
{
    return a+b+c;
}

int main ()
{
  int nResult, *pTemp;
  nResult = add(1,2,3);
  printf ("%d \r\n", nResult);
}
```



Pointer to a function

- every function has a base address attached to it
  - This base address acts as an entering point into that function.
  - this address can be stored in a specially designed pointer known as function pointer.
  - Address of the function can be obtained by using pointer name without parenthesis

- Once stored, a pointer to a function can be very useful:
  - to call that function
  - pass function as arguments to other functions as if it was a variable.

Pointer to a function - declaration

How to declare a pointer to a function?

return_type (* pointer_variable) (variable1_type variable1_name , variable2_type variable2_name,....);

How to initialize a pointer to a function?

pointer_variable = function-name;

Pointer to a function – example 1

```
#include <stdio.h>

int add (int a, int b, int c)
{
 return a+b+c;
}

int main () {
  int (*add_pointer) (int,int,int); //Declaration of function pointer for storage
     of base address of function add.
 add_pointer = add;
  printf ("%d \r\n", add_pointer(1,2,3));     //Print 6 as output
  return 0;
}
```