COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 7 - Class 1: Pointers - Part II

Administrivia

- *Small Programming Assignment 2* and *Large Programming Assignment 1* are out!!
- All assignments except SPA One have been returned.
  - SPA one grades will be available today.
- Exams grades have been released.

Today's Agenda

1. Quick Recap of past concepts
2. More on Pointers!

Quick Review

What are Pointers?

- Pointers are variables that store the address of a memory cell that contains a certain data type.
- * indicates that variable holds a memory location of certain type
- & is the address

```
int m = 25; // stored in address AA0

    int *itemp = &m;
```

Stack	Space
AA3	
AA2	
AA1	itemp = AA0
AA0	m = 25

The Dreference Operator *

- We have seen so far in this course that everything is stored somewhere in memory.
- Each memory has its own unique address.
- The pointer variable holds the specific address.
- The dereference operator acts like a “magic key” that allows access to the value stored.
- * is known as deference in C.

The Address Operator &

- We have been using & in our programs ever since scanf was introduced.
- & means address of
- Holds a value in hexadecimal that represents the location in memory.
  - This done with the placeholder `%p`.
  - Hexadecimal is a base 16 number. This means there are 16 unique digits.
- Think about it. Every time we used `scanf(“%d”, &num)` we were telling the compiler to store the value at the *Memory Address* of the variable named num.

More Pointers!

The Pointer Placeholder %p

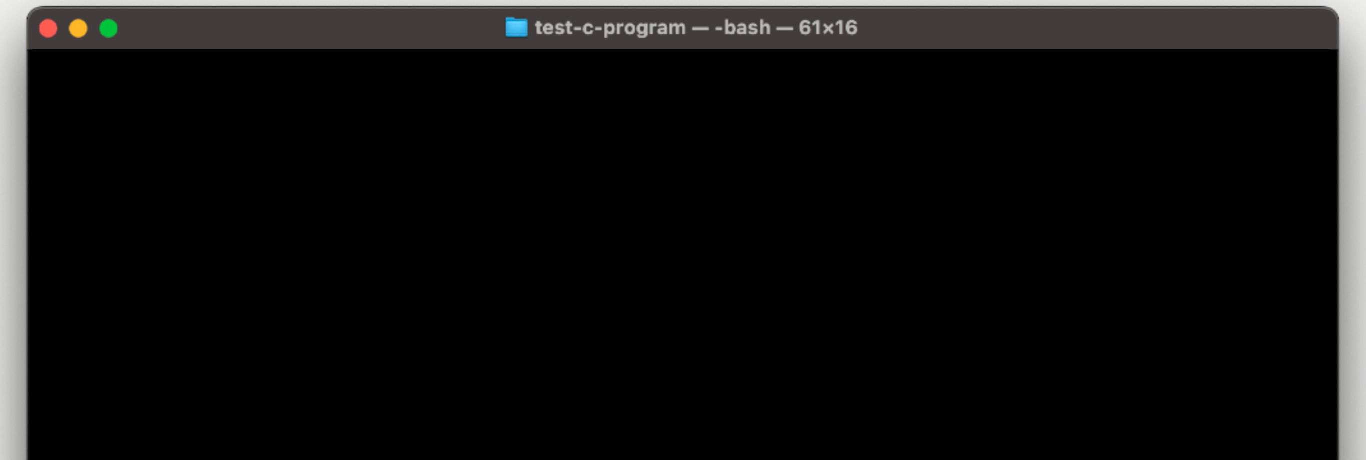
- There exists a special placeholder that can display the memory address of a reference.

```
int m = 25; // stored in address AA0

int *itemp = &m;

printf("The address of m is %p\n", &m);
printf("The address of itemp is %p\n", &itemp);
printf("itemp holds the value %p\n", itemp);
```

Displaying Address Example

Here →

```
int m = 25; // stored in address AA0

int *itemp = &m;

printf("The address of m is %p\n", &m);
printf("The address of itemp is %p\n", &itemp);
printf("itemp holds the value %p\n", itemp);
```

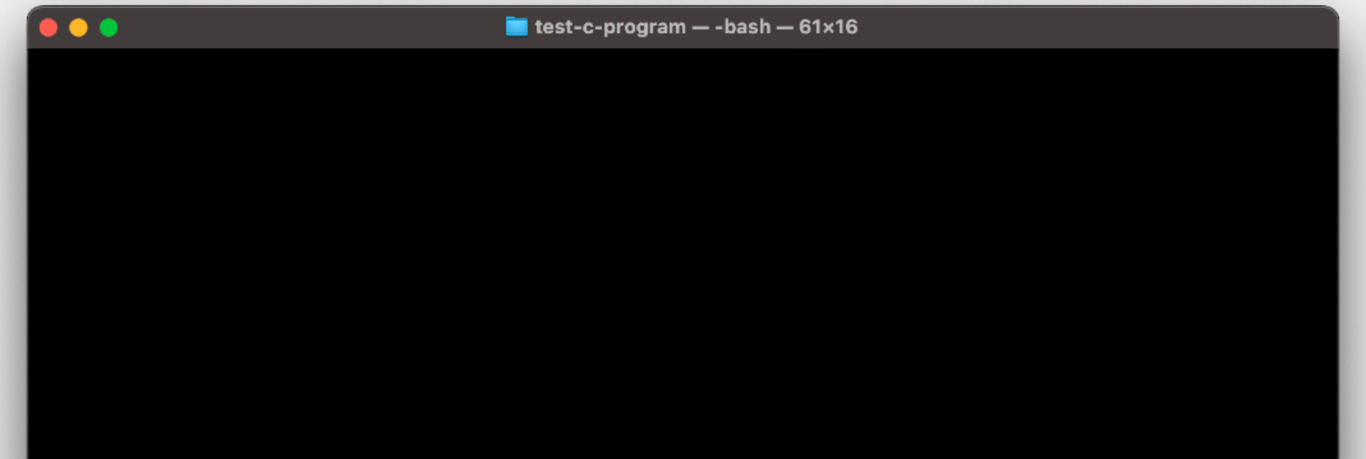


Stack	Space
AA3	
AA2	
AA1	
AA0	m = 25

Displaying Address Example

```
int m = 25; // stored in address AA0

int *itemp = &m;

printf("The address of m is %p\n", &m);
printf("The address of itemp is %p\n", &itemp);
printf("itemp holds the value %p\n", itemp);
```

Here →

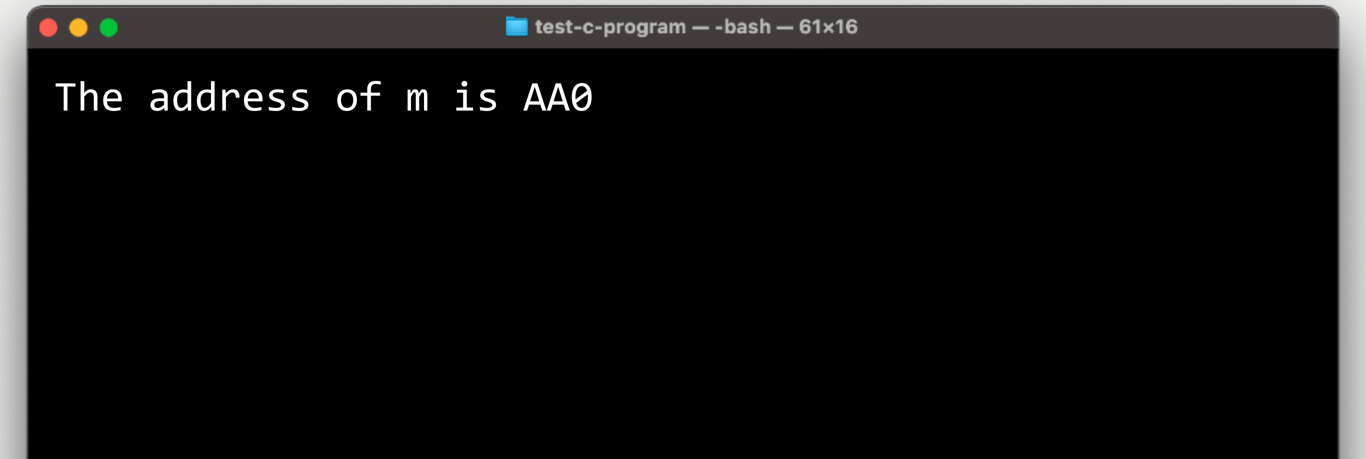


Stack	Space
AA3	
AA2	
AA1	itemp = AA0
AA0	m = 25

Displaying Address Example

```c
int m = 25; // stored in address AA0

int *itemp = &m;

printf("The address of m is %p\n", &m);
printf("The address of itemp is %p\n", &itemp);
printf("itemp holds the value %p\n", itemp);
```

Here



test-c-program — -bash — 61×16

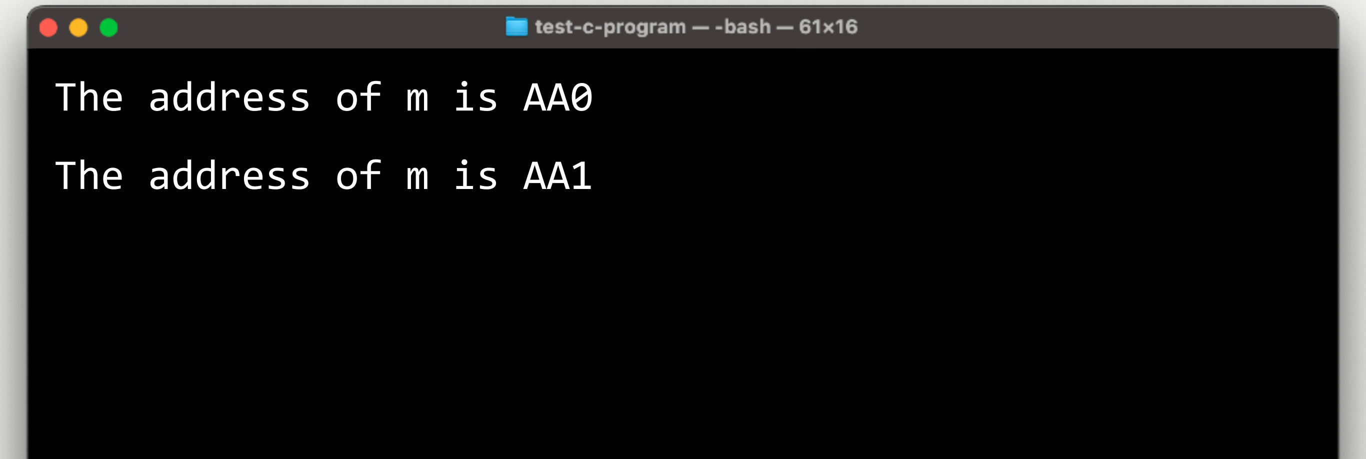
```
The address of m is AA0
```

Stack	Space
AA3	
AA2	
AA1	itemp = AA0
AA0	m = 25

Displaying Address Example

```c
int m = 25; // stored in address AA0

int *itemp = &m;

printf("The address of m is %p\n", &m);
printf("The address of itemp is %p\n", &itemp);
printf("itemp holds the value %p\n", itemp);
```

Here →



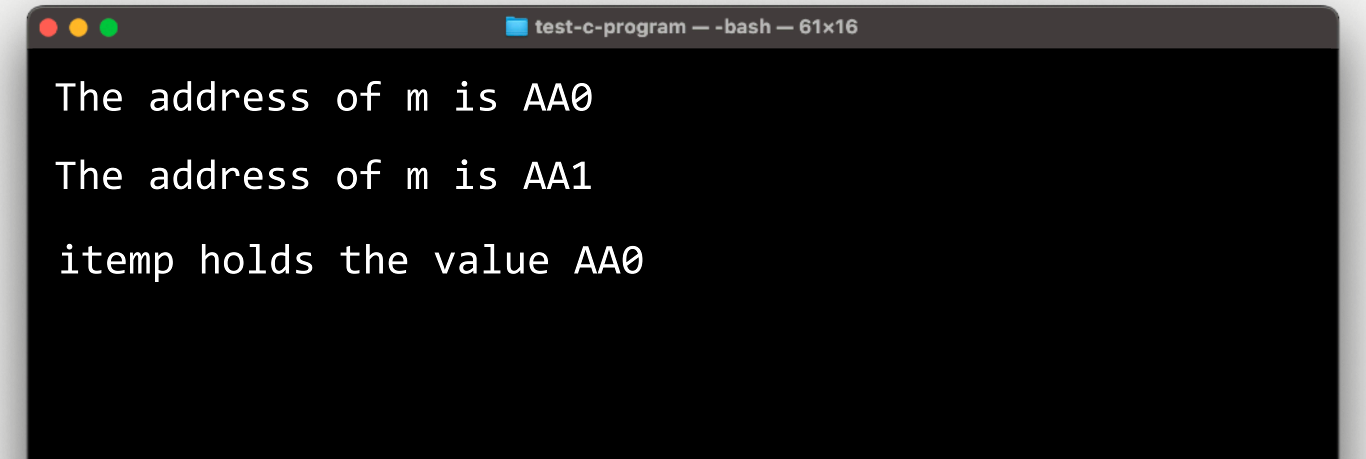
```
The address of m is AA0
The address of m is AA1
```

Stack	Space
AA3	
AA2	
AA1	itemp = AA0
AA0	m = 25

Displaying Address Example

```c
int m = 25; // stored in address AA0

int *itemp = &m;

printf("The address of m is %p\n", &m);
printf("The address of itemp is %p\n", &itemp);
printf("itemp holds the value %p\n", itemp);
```

Here →

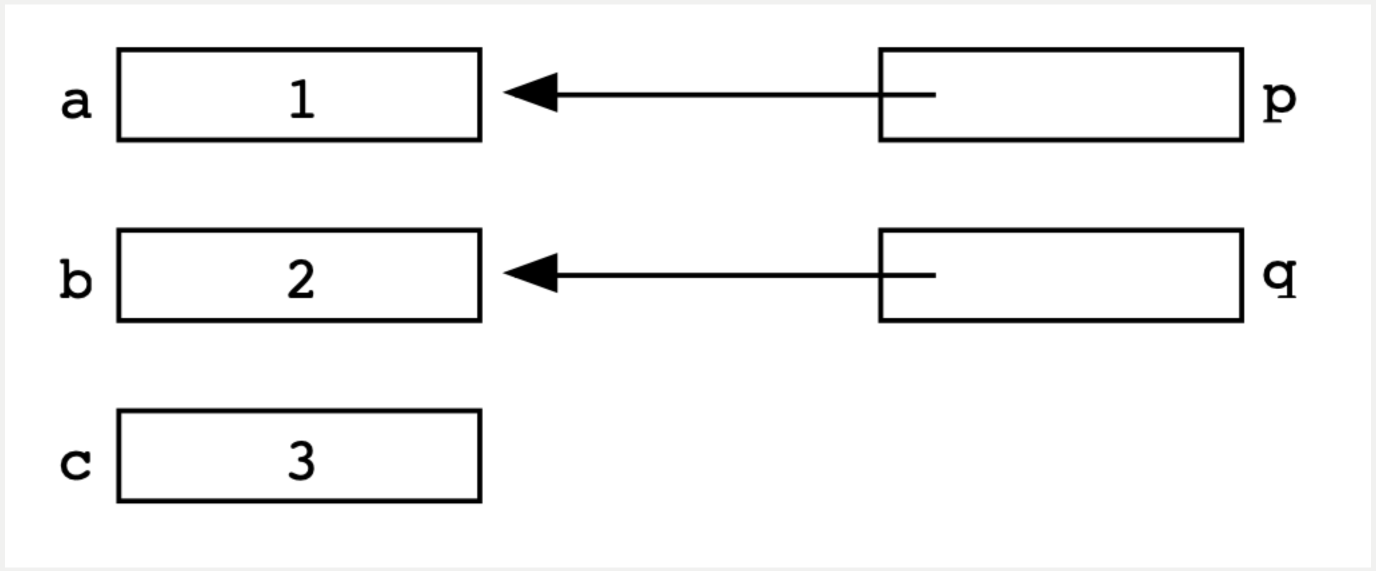


```
The address of m is AA0
The address of m is AA1
itemp holds the value AA0
```

Stack	Space
AA3	
AA2	
AA1	itemp = AA0
AA0	m = 25

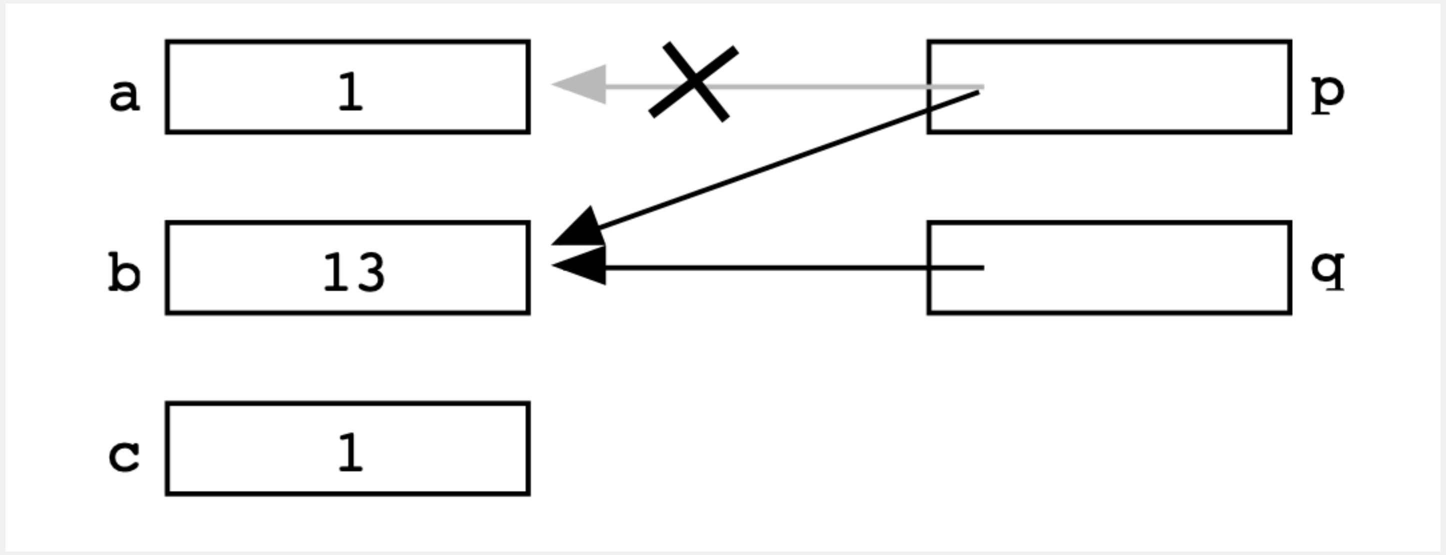
Pointer Example

```
int a = 1;
    int b = 2;
    int c = 3;
    int *p;
    int *q;

    p = &a; // set p to refer to a
    q = &b; // set q to refer to b
```



Pointer Example

```
int a = 1;
 int b = 2;
 int c = 3;
 int *p;
 int *q;

 p = &a; // set p to refer to a
 q = &b; // set q to refer to b

 c = *p; // retrieve p's pointee value (1) and put it in c
 p = q;  // change p to share with q (p's pointee is now b)
 *p = 13;  // dereference p to set its pointee (b) to 13 (*q is now 13)
```



The NULL/NIL Value

- Pointers that we have seen hold an address.
- Can pointers hold a value that doesn't represent an address in memory?
  - The simple answer is YES!
- NULL (or NIL) is a special value that represents nothing.
- We will see more of the value NULL being utilized when discussing dynamic memory.

```
int *ptr = NULL;
```

Stack	Space
AA3	
AA2	
AA1	
AA0	ptr = NULL

Functions with Parameters

- In past sessions, we have seen that variables have been passed by value.
- With pointers, we can now past variables by reference.
- Instead of making a local copy for the function, we can pass the memory location and perform computation on the variable in its original location. This is known as pass-by-reference.

Review: Pass By Value

Pass By Value Example

Here →

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	
AA0	

Pass By Value Example

Here

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	
AA0	num1 = 3

Pass By Value Example

Here

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	num2 =2
AA0	num1 = 3

Pass By Value Example

Here

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here →

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here →

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

Here

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here →

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

The values stored in num1, num2 and num3 are going to be copied respectively

Here

```
#include<stdio.h>

     nction (int numl, int num2, int num3);

     )

     = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```c
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here

Notice the parameters of the function are also num1, num2, and num3

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf ("num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here →

The values of num1, num2, and num3 are going to be copied and stored respectively with the provided parameters

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num1 = 3
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

Here

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 2
AA3	num1 = 3
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 2
AA3	num1 = 3
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold

Here

The value in red text is now the variable being modified

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 2
AA3	num1 = 5
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf ("num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold → `myFunction (num1, num2, num3);`

Here → `num2 = 8;`

The value in red text is now the variable being modified

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 8
AA3	num1 = 5
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf ("num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold →

Here →

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 8
AA3	num1 = 5
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 8
AA3	num1 = 5
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

Now we have reached the end of the user-defined function!

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold → myFunction (num1, num2, num3);

Here → printf ("num3 = %d\n", num3);

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	num3 = 1
AA4	num2 = 8
AA3	num1 = 5
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Hold (arrow at `myFunction (num1, num2, num3);`)

Here (arrow at `printf ("num3 = %d\n", num3);` in myFunction)

Now we have reached the end of the user-defined function!

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	~~num3 = 1~~
AA4	~~num2 = 8~~
AA3	~~num1 = 5~~
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here

After the function is done, it’s variables/ parameters are removed from the stack space.

Stack Space

AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n"  , num2);
printf ("num3 = %d\n", num3);
}
```

Here

After the function is done, it’s variables/ parameters are removed from the stack space.

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```c
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here

After the function is done, it’s variables/ parameters are removed from the stack space.

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

Pass By Value Example

```
#include<stdio.h>

void myFunction (int numl, int num2, int num3);

int main()
{
int num1 = 3;
int num2 = 2;
int num3 = 1;
printf (“num1 = %d\n", num1);
printf ("num2 = %d\n", num2);
printf ("num3 = %d\n", num3);

myFunction (num1, num2, num3);

printf ("num1 = %d\n", num1);
printf (“num2 = %d\n", num2) ;
printf ("num3 = %d\n", num3);
return 0;
}


void myFunction (int num1, int num2, int num3)
{
num1 = 5;
num2 = 8;

printf ("num1 = %d\n", num1);
printf ("num2 = %d\n" , num2);
printf ("num3 = %d\n", num3);
}
```

Here

After the function is done, it’s variables/ parameters are removed from the stack space.

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	num3 = 1
AA1	num2 = 2
AA0	num1 = 3

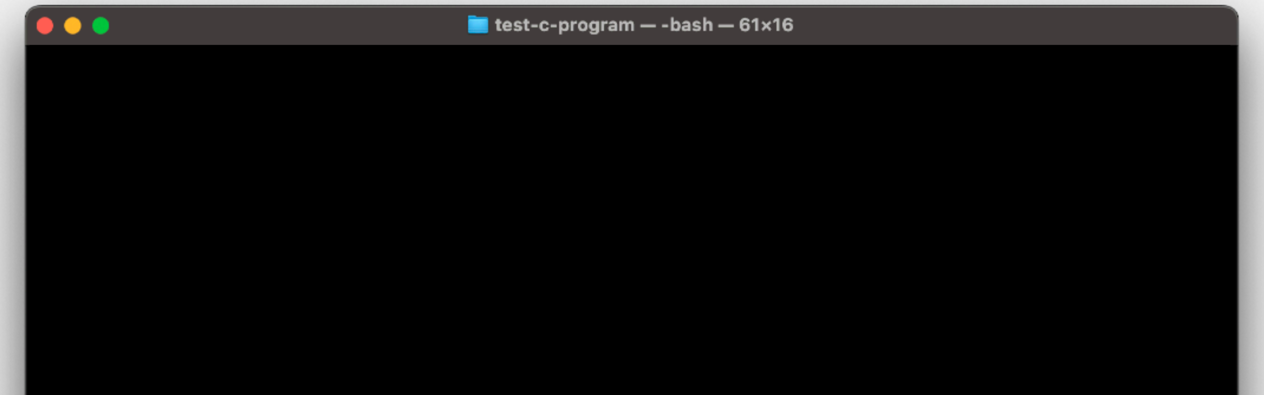
Pass By Reference (kinda) Example

Pass By "Reference" Example

Here

```
#include <stdio.h>

void increaseValue(int *num);

int main(void){

   int num = 13;

    printf("num = %d\n", num);

    increaseValue(&num);

    printf("num = %d\n", num);

    return 0;

}

void incraseValue(int *num){
    *num = *num + 1;
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	
AA0	

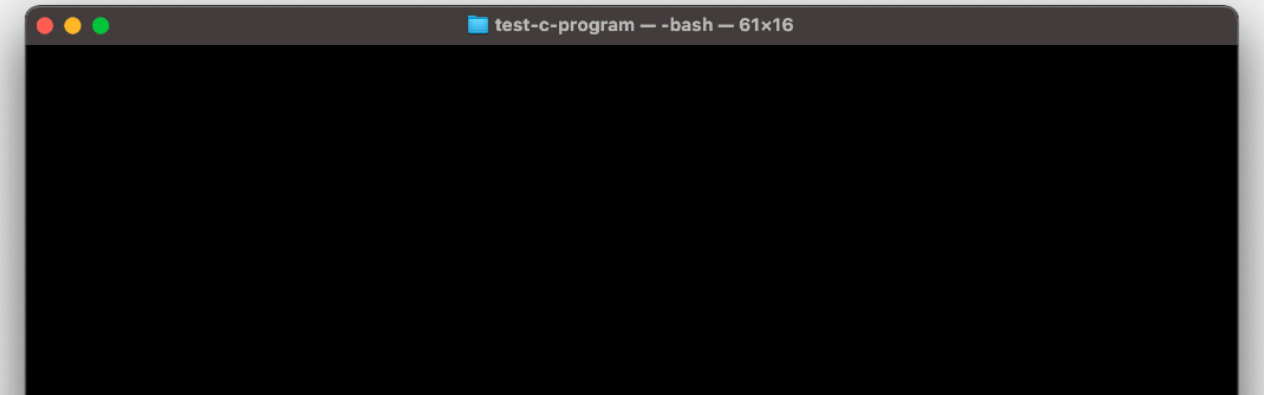


Pass By "Reference" Example

Here

```
#include <stdio.h>

void increaseValue(int *num);

int main(void){

   int num = 13;

    printf("num = %d\n", num);

    increaseValue(&num);

    printf("num = %d\n", num);

    return 0;

}

void incraseValue(int *num){
    *num = *num + 1;
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	
AA0	

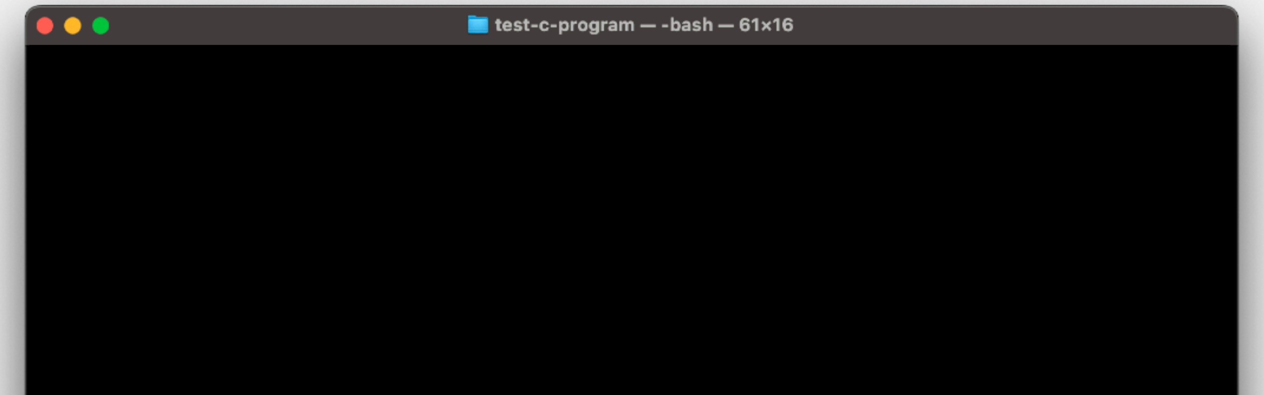


Pass By "Reference" Example

Here

```
#include <stdio.h>

void increaseValue(int *num);

int main(void){

   int num = 13;

    printf("num = %d\n", num);

    increaseValue(&num);

    printf("num = %d\n", num);

    return 0;

}

void incraseValue(int *num){
    *num = *num + 1;
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	
AA0	num = 13

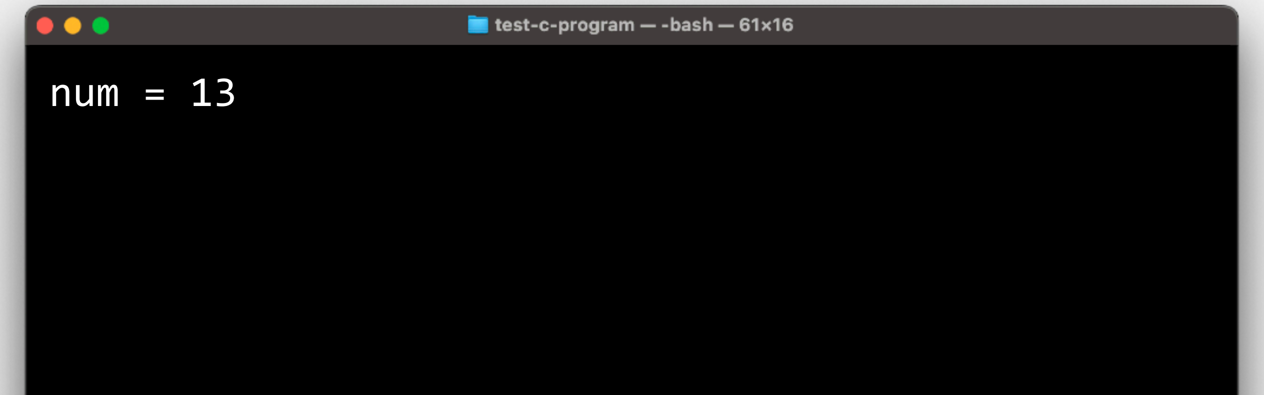


Pass By "Reference" Example

```c
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

    printf("num = %d\n", num);

    increaseValue(&num);

    printf("num = %d\n", num);

    return 0;

}

void incraseValue(int *num){
    *num = *num + 1;
}
```

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	
AA2	
AA1	
AA0	num = 13



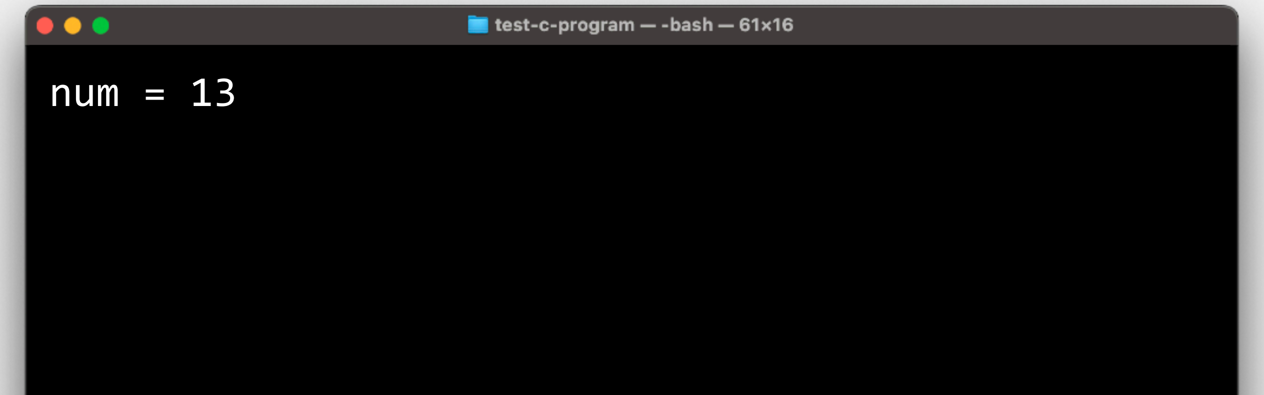
```
num = 13
```

Pass By "Reference" Example

```
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

     printf("num = %d\n", num);

     increaseValue(&num);

     printf("num = %d\n", num);

     return 0;

}

void incraseValue(int *num){
     *num = *num + 1;
}
```

Here →

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num = AA0
AA2	
AA1	
AA0	num = 13



```
num = 13
```

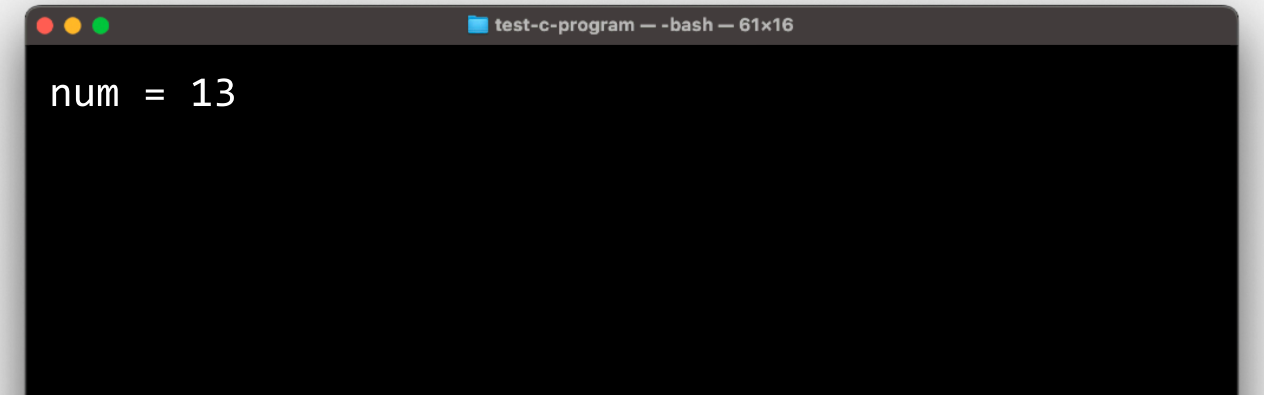
Pass By "Reference" Example

```
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

     printf("num = %d\n", num);

     increaseValue(&num);

     printf("num = %d\n", num);

     return 0;

}

void incraseValue(int *num){
    *num = *num + 1;
}
```

Hold

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num = AA0
AA2	
AA1	
AA0	num = 13



```
num = 13
```

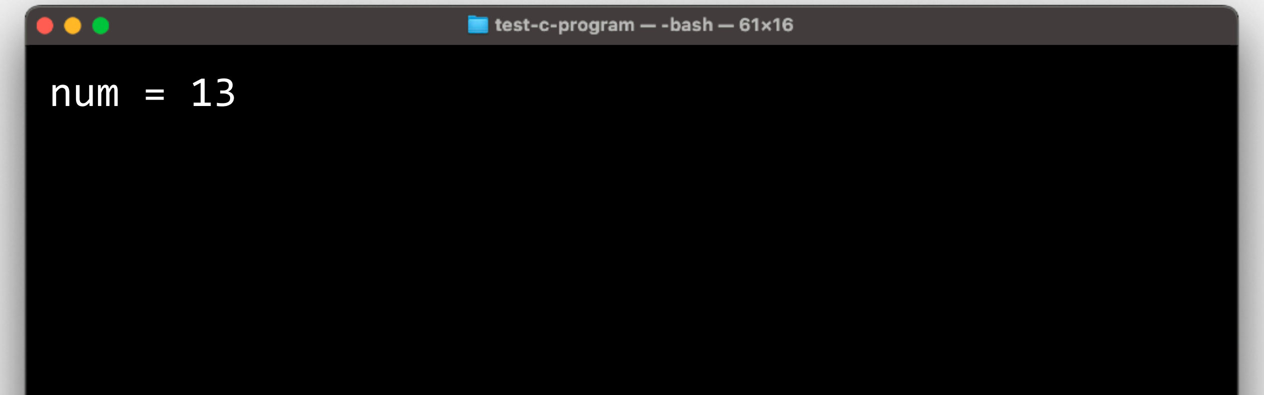
Pass By "Reference" Example

```c
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

     printf("num = %d\n", num);

     increaseValue(&num);

     printf("num = %d\n", num);

     return 0;

}

void incraseValue(int *num){
     *num = *num + 1;
}
```

Hold

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num = AA0
AA2	
AA1	
AA0	num = 14



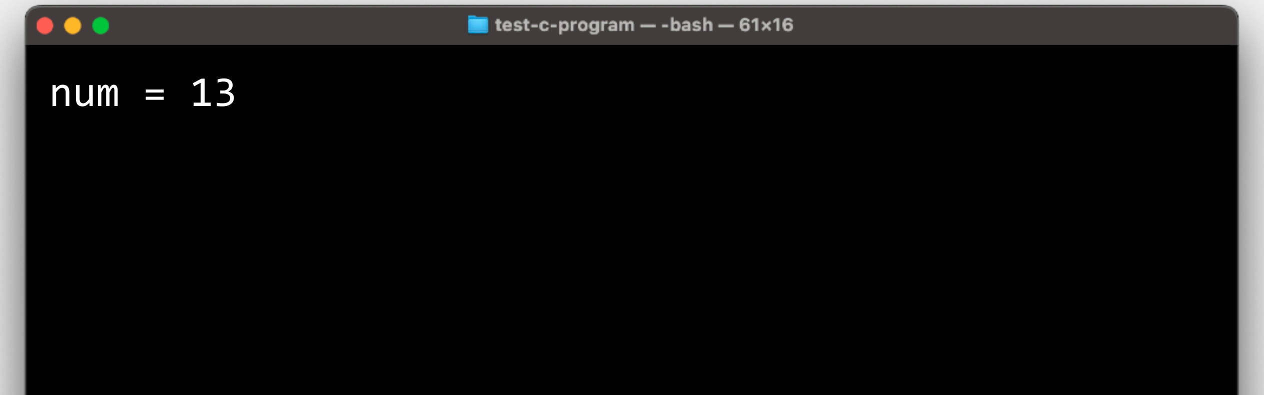
```
num = 13
```

Pass By "Reference" Example

Here

```c
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

     printf("num = %d\n", num);

     increaseValue(&num);

     printf("num = %d\n", num);

     return 0;

}

void incraseValue(int *num){
     *num = *num + 1;
}
```

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num = AA0
AA2	
AA1	
AA0	num = 14



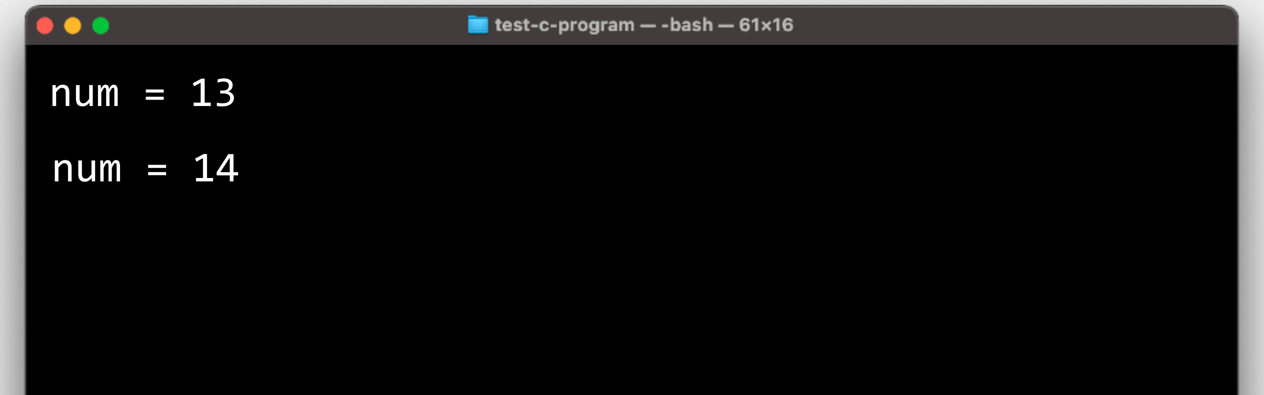
```
num = 13
```

Pass By "Reference" Example

```c
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

     printf("num = %d\n", num);

     increaseValue(&num);

     printf("num = %d\n", num);

     return 0;

}

void incraseValue(int *num){
     *num = *num + 1;
}
```

Here

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num = AA0
AA2	
AA1	
AA0	num = 14



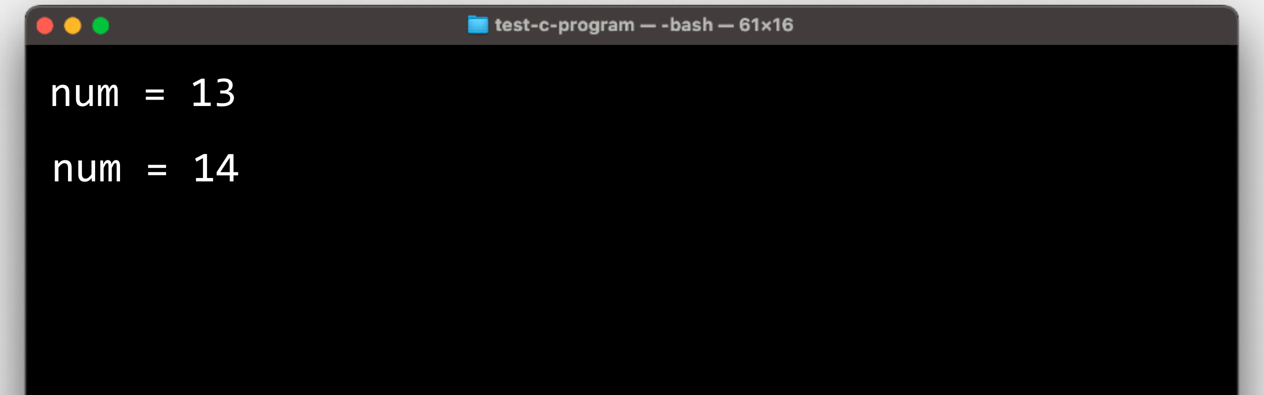
```
num = 13
num = 14
```

Pass By "Reference" Example

```c
#include <stdio.h>

void increaseValue(int *num);

int main(void){

    int num = 13;

     printf("num = %d\n", num);

     increaseValue(&num);

     printf("num = %d\n", num);

     return 0;

}

void incraseValue(int *num){
     *num = *num + 1;
}
```

Here →

Stack Space	
AA9	
AA8	
AA7	
AA6	
AA5	
AA4	
AA3	num = AA0
AA2	
AA1	
AA0	num = 14



```
num = 13
num = 14
```

Scope of Names

- Scope of a name refers to the region in a program where a particular meaning of a name is visible.
- Local and Global Variables
- When variables are being used, certain functions may not be able to access them due to where they were declared!
- Why can't everything be global? Would that be easier?

```c
#include <stdio.h>

void increaseValue(int *num);
void calculate();

int var; // global variable BAD!!

int main(void){

   int num = 13;

    printf("num = %d\n", num);

return 0;

}

void calculate(){

    int num1;   // local variable
    int num2;   // local variable
    scanf("%d%d", &num1, &num2);

    int result = num1 + num2;

}
```

Acknowledgements

Slides adapted from Dr. Andrew Steinberg’s COP 3223H course