Review

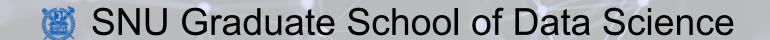
- Compared to Python, a few more things need to be considered to play with functions in C
 - You need to write **data types** of its parameters and return value explicitly in the function header
 - You can **declare** a function using its prototype before its full definition

- C uses stack to manage memory operation of a function call
 - We have seen why using stack (LIFO data structure) makes sense
 - We have seen how all the information is pushed and popped correctly using only the two pointers (stack pointer and frame pointer)

Pointers - Motivation

Lecture 29-1

Hyung-Sin Kim



Practice – Call by Value

Swapping function in C

```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7:
  int valB = 5:
  printf("Before Swap: valA = \%d, valB = \%d\n", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0:
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```

Let's type, compile, and execute!

What do you see on your screen?

Does swapping happen as intended?

Can you explain why you see such results by using memory operation we learned last time?

Practice – Call by Value (Solution)

Swapping function in C

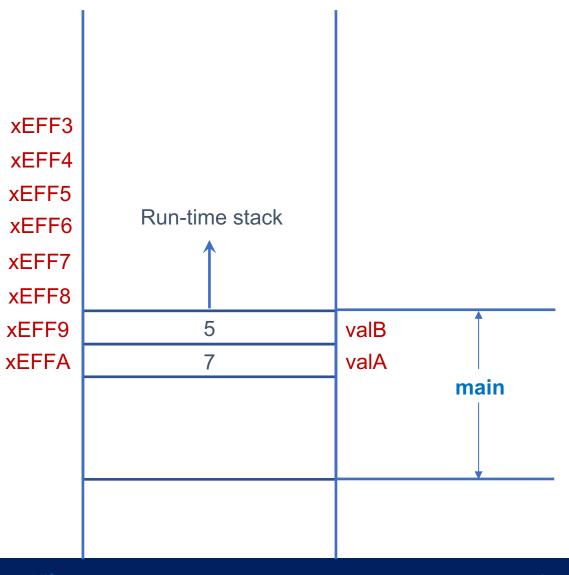
```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7:
  int valB = 5:
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = \%d, valB = \%d\n", valA, valB);
  return 0:
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```

Local variables firstVal and secondVal **die** when function swap ends!

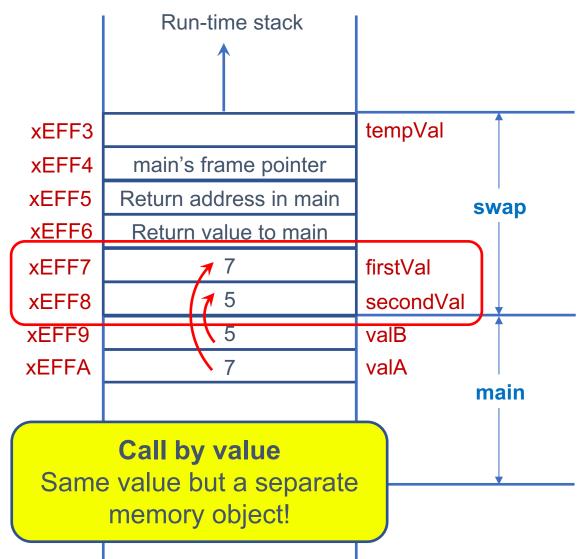
Swapping does not happen!
(well... not surprising because swap function does not return anything)

Let's see what happens in the memory

```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d\n", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = %d, valB = %dn", valA, valB);
  return 0;
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = \%d, secondVal = \%d\n", firstVal, secondVal);
```

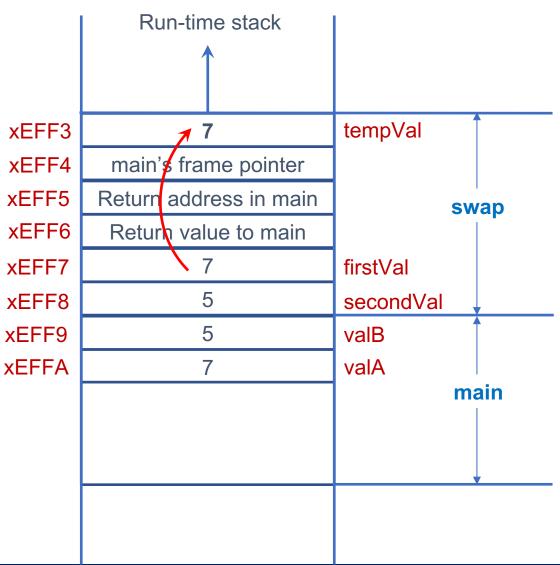


```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```



Swapping function in C

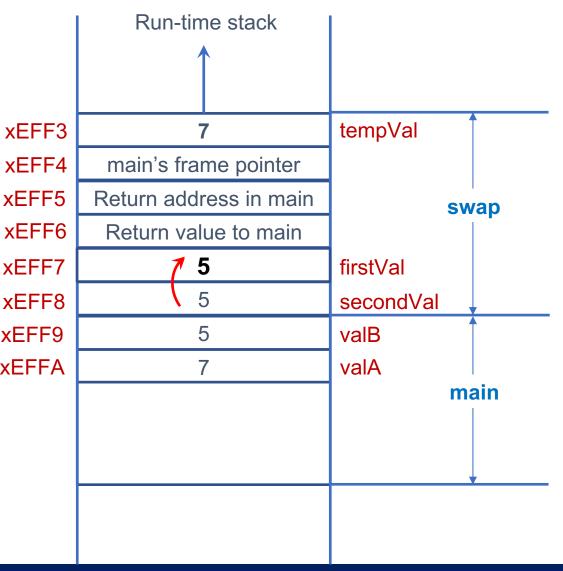
```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = \%d, valB = \%d", valA, valB);
  return 0;
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```



xEFF9

Swapping function in C

```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = \%d, valB = \%d", valA, valB);
  return 0;
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal;
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```



xEFF3

xEFF5

xEFF7

xEFF8

xEFF9

Swapping function in C

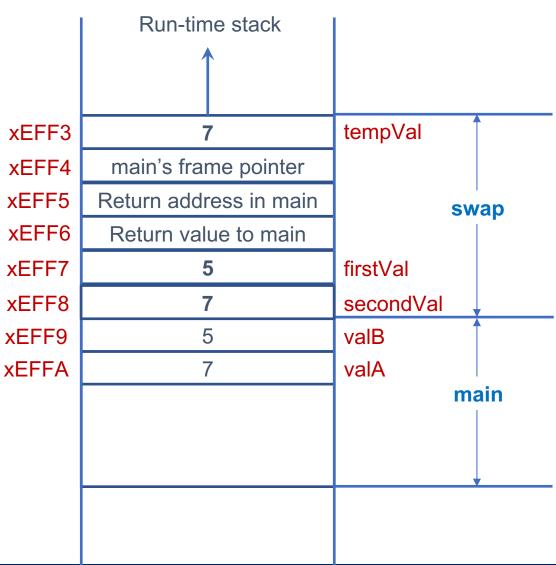
```
#include <stdio.h>
                                                                                                                tempVal
                                                                     xEFF3
void swap(int firstVal, int secondVal);
                                                                     xEFF4
                                                                                  main's frame pointer
int main(void) {
                                                                     xEFF5
                                                                                 Return address in main
 int valA = 7;
                                                                                                                              swap
                                                                     xEFF6
                                                                                  Return value to main
 int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
                                                                     xEFF7
                                                                                              5
                                                                                                                firstVal
  swap(valA, valB);
                                                                     xEFF8
                                                                                                                secondVal
  printf("After Swap: valA = \%d, valB = \%d", valA, valB);
  return 0;
                                                                     xEFF9
                                                                                              5
                                                                                                                valB
                                                                     xEFFA
                                                                                                                valA
                                                                                                                              main
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
                                                    In Swap: firstVal = 5, secondVal = 7
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = \%d, secondVal = \%d\n", firstVal, secondVal);
```

10

Run-time stack

Swapping function in C

```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```



11

Swapping function in C

```
#include <stdio.h>
void swap(int firstVal, int secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  swap(valA, valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
    After Swap: valA = 7, valB = 5
void swap(int firstVal, int secondVal) {
  int tempVal;
  tempVal = firstVal;
  firstVal = secondVal:
  secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", firstVal, secondVal);
```

All local variables (swap scope) are gone! Run-time stack 5 valB valA main

xEFF3

xEFF4

xEFF5

xEFF6

xEFF7

xEFF8

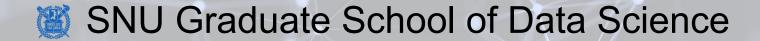
xEFF9

Computing Bootcamp

Pointers

Lecture 29-2

Hyung-Sin Kim



How can we make the swap function impact the arguments?

Pointer – Declaration

- A pointer variable contains an **address** of a memory object (e.g., variable)
 - <type> *<name>
 - int *ptr; ptr is a variable that contains an address of an integer variable
 - char *ptr;

 ptr is a variable that contains an address of a character variable
- Address operator & and indirection operator *
 - int intVariable = 10; // Assume that intVariable's address is 0xEE01
 - int *intPtr;
 - intPtr = &intVariable;
 - Now intPtr contains intVariable's address
 - *intPtr is the value in the memory object that intPtr points to (i.e., intVariable's value, 10)
 - "*intPtr = *intPtr + 2" is the same as "intVariable = intVariable + 2"



15

Let's print some values!

• Swapping function in C (pointer version)

```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = %d, valB = %d\n", valA, valB);
  return 0;
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```

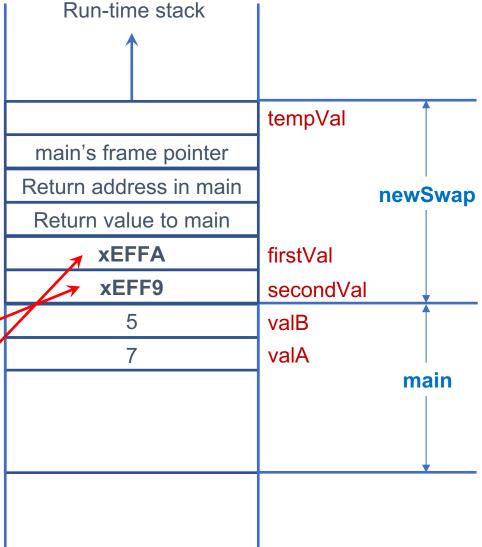
Let's type, compile, and execute!

What do you see on your screen?

Does swapping happen as intended?

Swapping function in C

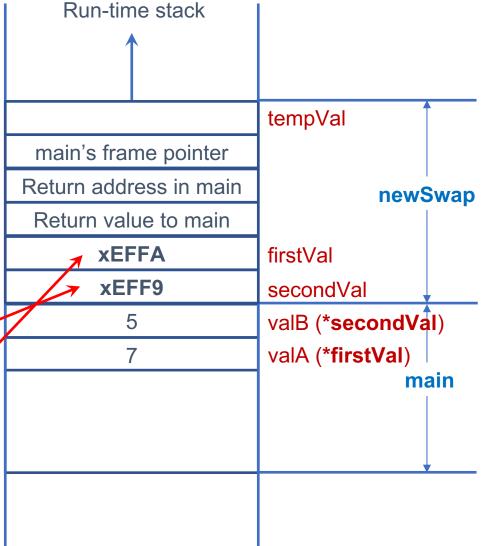
```
#include <stdio.h>
                                                                        xEFF3
void newSwap(int *firstVal, int *secondVal);
                                                                       xEFF4
int main(void) {
                                                                       xEFF5
  int valA = 7;
                                                                       xEFF6
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
                                                                       xEFF7
  newSwap(&valA, &valB);
                                                                       xEFF8
  printf("After Swap: valA = \%d, valB = \%d", valA, valB);
  return 0;
                                                                       xEFF9
                                                                       xEFFA'
void newSwap(int *firstVal, int *secondVal) {
                                                           Address!
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```



18

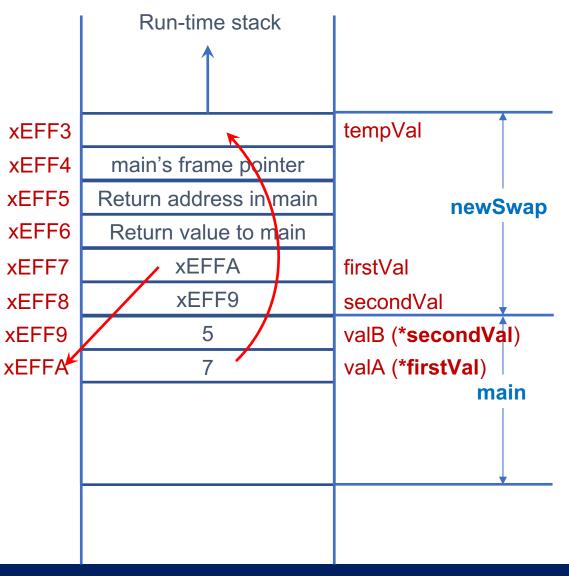
Swapping function in C

```
#include <stdio.h>
                                                                       xEFF3
void newSwap(int *firstVal, int *secondVal);
                                                                       xEFF4
int main(void) {
                                                                       xEFF5
  int valA = 7;
                                                                       xEFF6
  int valB = 5;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
                                                                       xEFF7
  newSwap(&valA, &valB);
                                                                       xEFF8
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
                                                                      xEFF9
                                                                      xEFFA
void newSwap(int *firstVal, int *secondVal) {
                                                           Address!
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```

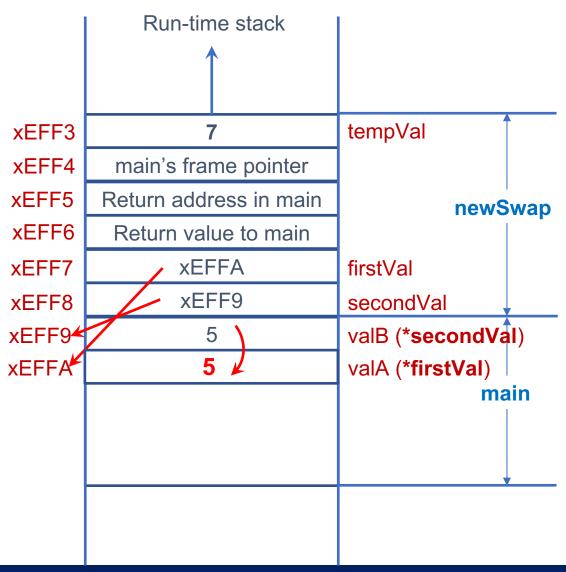


19

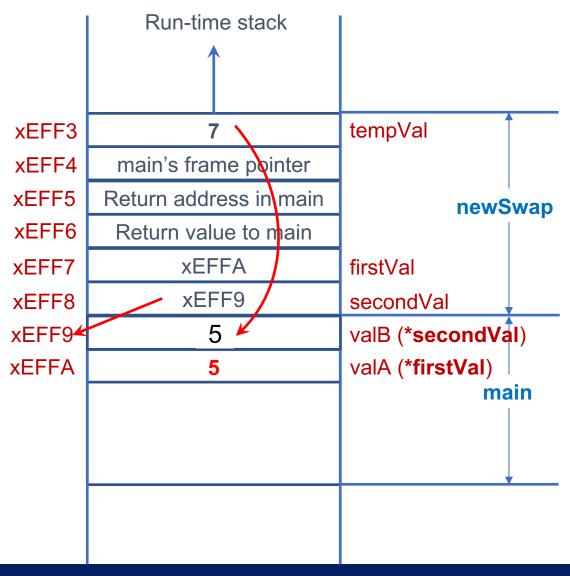
```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 7;
  int valB = 5;
  printf("Before Swap: valA = %d, valB = %d", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```



```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 3;
  int valB = 4;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal;
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```



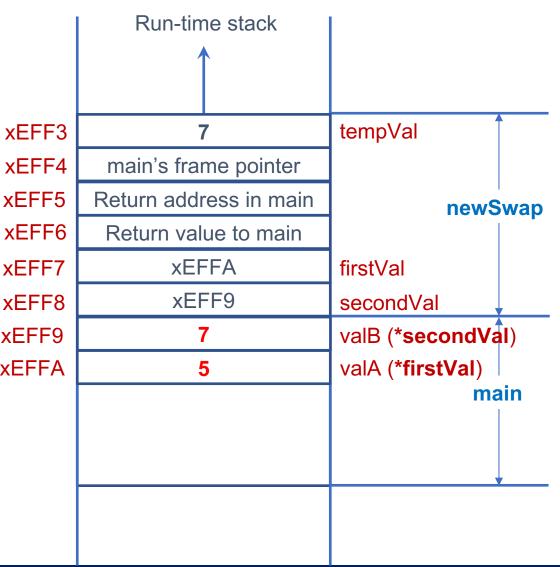
```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 3;
  int valB = 4;
  printf("Before Swap: valA = \%d, valB = \%d\n", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = %d, valB = %d", valA, valB);
  return 0;
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```



Run-time stack Swapping function in C #include <stdio.h> xEFF3 tempVal void newSwap(int *firstVal, int *secondVal); xEFF4 main's frame pointer int main(void) { xEFF5 Return address in main newSwap int valA = 3; xEFF6 Return value to main int valB = 4; printf("Before Swap: valA = %d, $valB = \%d\n$ ", valA, valB); xEFF7 xEFFA firstVal newSwap(&valA, &valB); xEFF9 xEFF8 secondVal printf("After Swap: valA = %d, valB = %d", valA, valB); return 0; valB (*secondVal) xEFF9⁴ xEFFA valA (*firstVal) main void newSwap(int *firstVal, int *secondVal) { int tempVal; tempVal = *firstVal; In Swap: firstVal = 5, secondVal = 7 *firstVal = *secondVal: *secondVal = tempVal; printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);

Swapping function in C

```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 3;
  int valB = 4;
  printf("Before Swap: valA = \%d, valB = \%d\n", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = \%d, valB = \%d\n", valA, valB);
  return 0;
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```



xEFF4

xEFF5

xEFF7

xEFF8

xEFF9

Swapping function in C

```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 3;
  int valB = 4;
  printf("Before Swap: valA = \%d, valB = \%d", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = \%d, valB = \%d\n", valA, valB);
  return 0;
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```

Local variables are gone. But valA and valB are changed! Run-time stack valB valA main

xEFF3

xEFF4

xEFF5

xEFF6

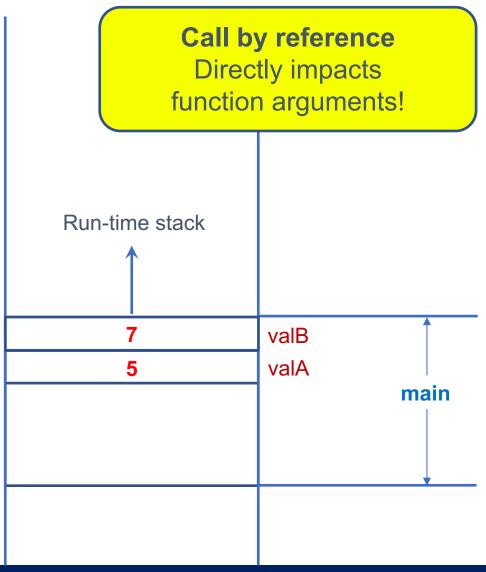
xEFF7

xEFF8

xEFF9

Swapping function in C

```
#include <stdio.h>
void newSwap(int *firstVal, int *secondVal);
int main(void) {
  int valA = 3;
  int valB = 4;
  printf("Before Swap: valA = \%d, valB = \%d\n", valA, valB);
  newSwap(&valA, &valB);
  printf("After Swap: valA = \%d, valB = \%d\n", valA, valB);
  return 0;
      After Swap: valA = 5, valB = 7
void newSwap(int *firstVal, int *secondVal) {
  int tempVal;
  tempVal = *firstVal;
  *firstVal = *secondVal:
  *secondVal = tempVal;
  printf("In Swap: firstVal = %d, secondVal = %d\n", *firstVal, *secondVal);
```



xEFF3

xEFF4

xEFF5

xEFF6

xEFF7

xEFF8

xEFF9

Pointer – etc.

- Null pointers A special case pointer that points to **nothing**
 - int *ptr;
 - ptr = NULL; // NULL is a specially defined preprocessor macro that contains a value 0
 - It is useful to initialize a pointer to NULL when it does not point anything yet
- Demystifying the syntax
 - Pointer declaration (e.g., int *ptr;)
 - Declaring a variable <u>ptr</u> that, when the <u>indirection operator *</u> is applied to it, generates a value of type <u>int</u>
 - That is, *ptr is integer type
 - Input library function scanf("%d", &input);
 - To change the value of the function argument "input," scanf must have the address of "input"
 - If you omit &, C compiler will kindly give an error message

Summary

- Pointer
 - Motivation
 - Declaration
 - Swap

Thanks!