Pointers and Dynamic Memory

CIS 308

Jorge Valenzuela

KANSAS STATE UNIVERSITY

1

Outline

- Pointers
  - Var Declaration
  - & and * Operators
  - Pointers and Arrays
  - Iteration with Pointers
  - Pointers to Pointers
  - Call by Reference

KANSAS STATE UNIVERSITY

2

Outline

- Dynamic Memory
  - Stack v. Heap
  - sizeof()
  - malloc()
  - calloc()
  - realloc()
  - free()

KANSAS STATE UNIVERSITY

3

Pointers

- Google Directions
  - Map
  - Lot/GPS Coord.
  - 1345 N Bluemont Av.
  - Peter's House
  - Duplex
  - House stuff
- Pointers Vocabulary
  - ?
  - ?
  - ?
  - ?
  - ?
  - ?

KANSAS STATE UNIVERSITY

4

Pointers

- Google Directions
  - Map
  - Lot
  - 1345 N Bluemont Av.
  - Peter's House
  - Duplex
  - House stuff
- Pointers Vocabulary
  - Memory
  - Mem. Location
  - Location Address
  - Location Name
  - Location Content Type
  - Location Content

KANSAS STATE UNIVERSITY

5

Pointers

- Declaration
  - Non Pointers
    - type varName
      - int x; *//How do I interpret what's store in x?*
  - Pointers vars
    - type * name
      - int * numPtr;
      - int * numPtr;
      - int *numPtr; *// How do I interpret what's*
      - *store in* numPtr*?*

KANSAS STATE UNIVERSITY

6

Pointers

- & and * Operators
  - & Returns the address of the variable
  - * Returns/accesses the content of the location address

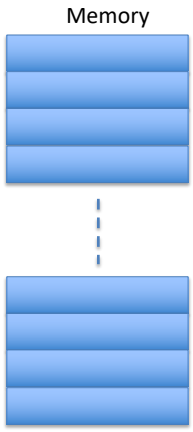
KANSAS STATE UNIVERSITY

7

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
```



KANSAS STATE UNIVERSITY

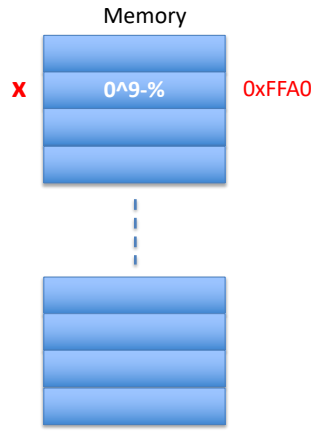
8

Pointers

- & and * Operators

  int x = 7;

  int * ptrX;



KANSAS STATE UNIVERSITY

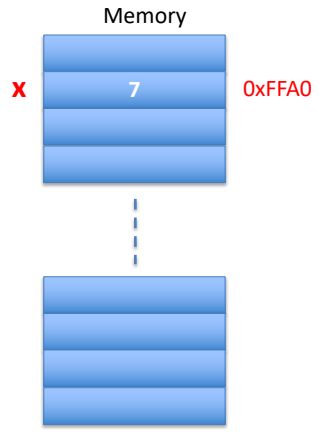
9

Pointers

- & and * Operators

  int x = 7;

  int * ptrX;



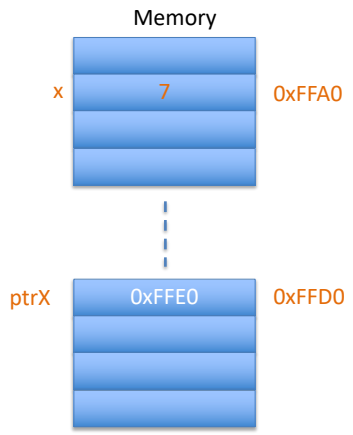
KANSAS STATE UNIVERSITY

10

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
```



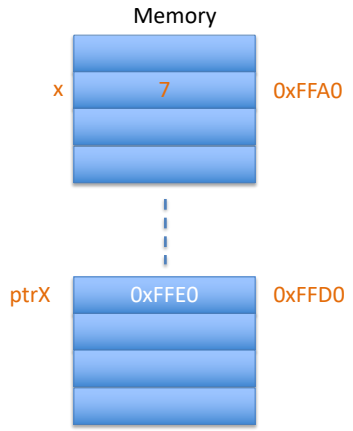
KANSAS STATE UNIVERSITY

11

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
```



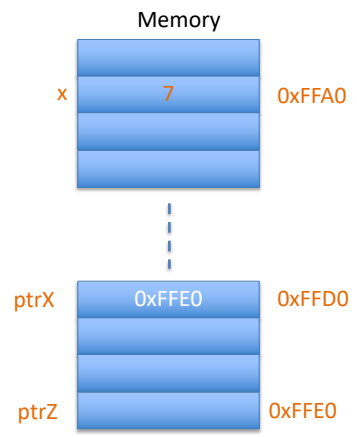
KANSAS STATE UNIVERSITY

12

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```



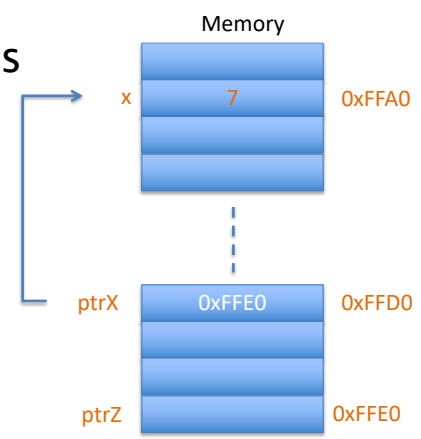
KANSAS STATE UNIVERSITY

13

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```



KANSAS STATE UNIVERSITY

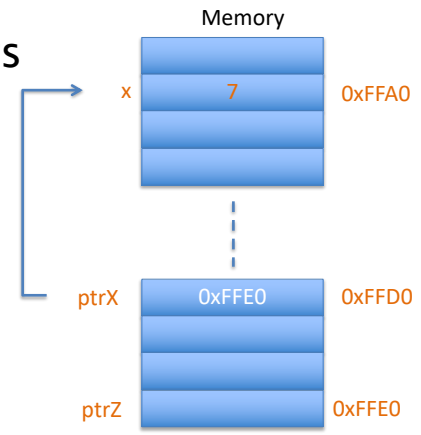
14

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```

```
ptrX = x;
*ptrX = &x;
ptrX = &x;
```



KANSAS STATE UNIVERSITY

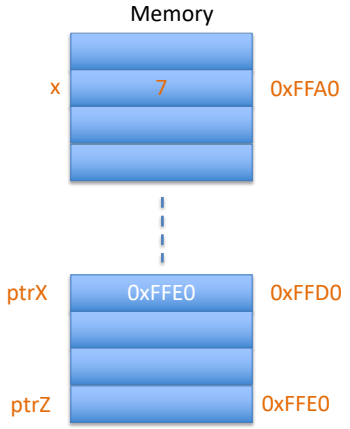
15

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```

```
ptrX = x;
*ptrX = &x;
ptrX = &x;
```



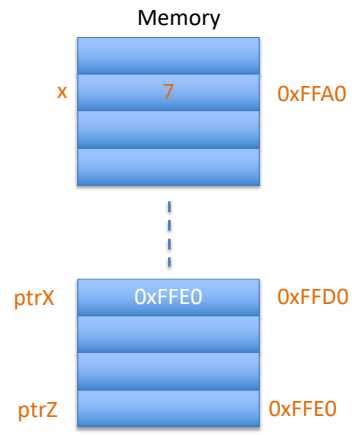
KANSAS STATE UNIVERSITY

16

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```




```
ptrX = x;
*ptrX = &x;
ptrX = &x;
```

```
pointers.c:12:7: warning: incompatible integer to pointer conversion assigning to
    'int *' from 'int'; take the address with & [-Wint-conversion]
  ptrX = x;
      ^ ~
      &
```

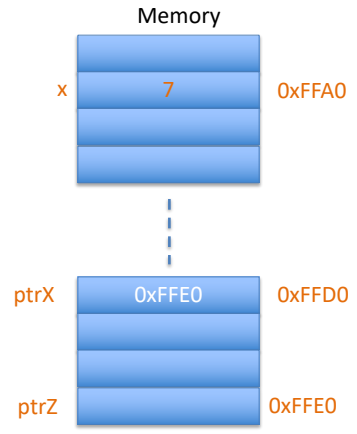

KANSAS STATE UNIVERSITY

17

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```



~~ptrX = x;~~

*ptrX = &x;

ptrX = &x;

KANSAS STATE UNIVERSITY

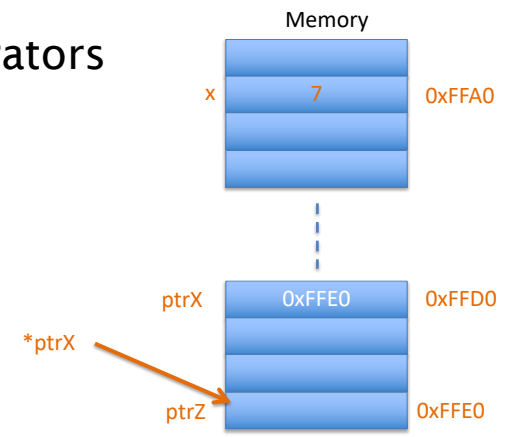
18

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```

~~ptrX = x;~~
*ptrX = &x;
ptrX = &x;



KANSAS STATE UNIVERSITY

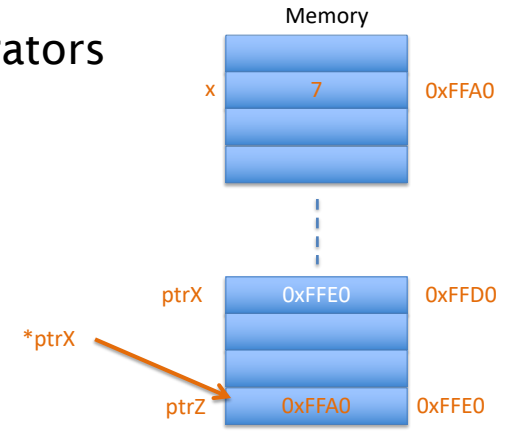
19

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```

~~ptrX = x;~~
*ptrX = &x;
ptrX = &x;



KANSAS STATE UNIVERSITY

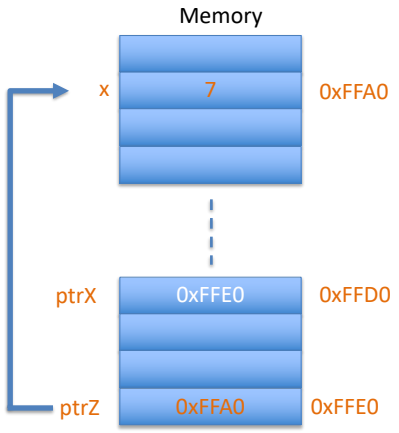
20

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```

~~ptrX = x;~~
*ptrX = &x;
ptrX = &x;



KANSAS STATE UNIVERSITY

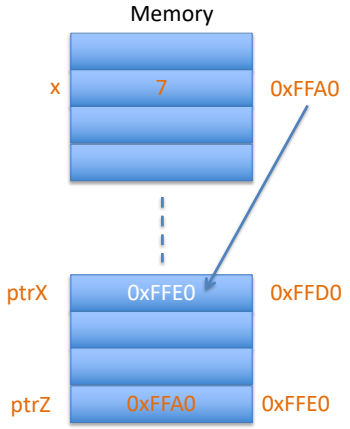
21

Pointers

- & and * Operators

```
int x = 7;
int *  ptrX;
int *  ptrZ;
```

~~ptrX = x;~~
~~*ptrX = &x;~~
ptrX = &x;

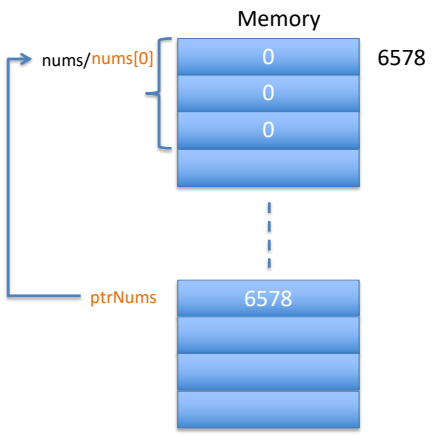


KANSAS STATE UNIVERSITY

22

Pointers and Arrays

- int nums[3]
- int *ptrNums;
- ptrNums = nums;



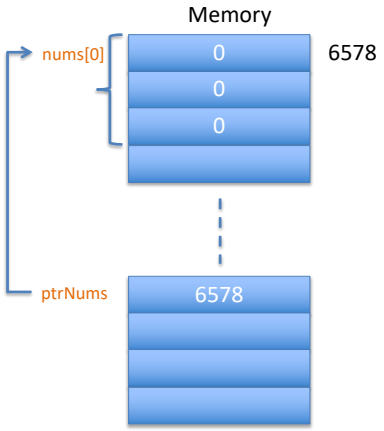
KANSAS STATE UNIVERSITY

23

Pointers and Arrays

- int nums[3]
- Int *ptrNums;
- ptrNums = nums;

*(ptrNums+2) = 7;

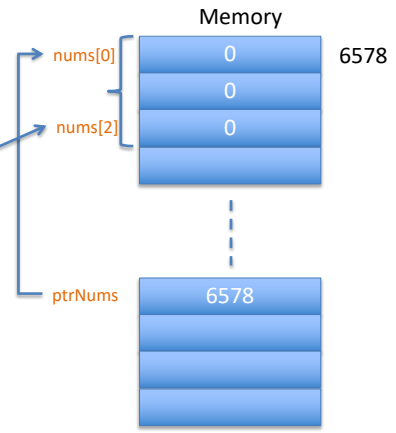


KANSAS STATE UNIVERSITY

24

Pointers and Arrays

- int nums[3]
- int *ptrNums;
- ptrNums = nums;



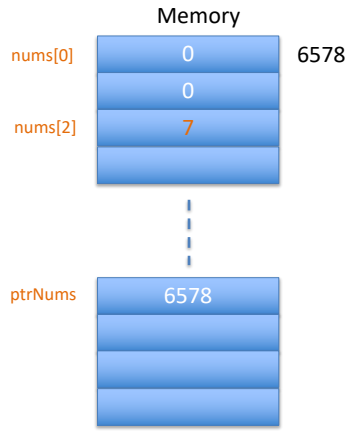
*(ptrNums+2)

KANSAS STATE UNIVERSITY

25

Pointers and Arrays

- int nums[3]
- int *ptrNums;
- ptrNums = nums;



*(ptrNums+2) = 7;

KANSAS STATE UNIVERSITY

26

Pointers and Arrays

- int nums[3]
- Int *ptr;

```
For(ptr = nums; ptr < nums+3; ptr++) {
    *ptr = 0;
}
```


KANSAS STATE UNIVERSITY

27

Pointers to Pointers

- int i = 7;
- int *ptr;
- int **pptr;

```
ptr = &i;        // ptr points to i
*ptr = 99;       // now i is 99

pptr = &ptr;     // pptr points to i
**pptr = 23;     // now i = 23
```


KANSAS STATE UNIVERSITY

28

Call by Reference

Call by Value

```
void swap(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

Call by Reference

```
void swap(int *a, int *b) {
    int temp = *a;
    *a = * b;
    *b = temp;
}
```


KANSAS STATE UNIVERSITY

29

Dynamic Memory

- Stack
  - Computer's memory that stores temporary variables created by each function (including the main() function).
  - The stack is a "LIFO" (last in, first out) data structure, that is managed and optimized by the CPU
  - **when a function exits**, all of its variables are popped off of the stack (and hence lost forever)
- Heap
  - A region of your computer's memory that is not managed automatically for you, and is not as tightly managed by the CPU
  - Allocate and deallocate the memory you use
  - Memory leaks

KANSAS STATE UNIVERSITY

30

Dynamic Memory

- Stack
  - Don't have to explicitly de-allocate vars
  - Local vars
  - More limited size (OS-dep)
  - No resizable vars
- Heap
  - Explicitly de-allocate vars
  - Global vars
  - Resizable vars

KANSAS STATE UNIVERSITY

31

Dynamic Memory

- Use of:
  - #include <stdlib.h>
- Use of:
  - malloc()
  - calloc()
  - realloc()
  - free()

KANSAS STATE UNIVERSITY

32

Dynamic Memory

- sizeof(type)

- int → 4
- double → 8
- int* → 4
- char → 1
- char* → 4

KANSAS STATE UNIVERSITY

33

Dynamic Memory

```
int nums1[5];
```


KANSAS STATE UNIVERSITY

34

Dynamic Memory

```
int nums1[5];
int *nums2 = malloc(5*sizeof(int));
```


KANSAS STATE UNIVERSITY

35

Dynamic Memory

```
int nums1[5];
int *nums2 = malloc(5*sizeof(int));
```

- calloc(5*sizeof(int));
- realloc(nums2 , 10*sizeof(int));
- free(nums2)

KANSAS STATE UNIVERSITY

36

Pointers and Dynamic Memory

Lab Activity

KANSAS STATE UNIVERSITY

37