CSC 211: Computer Programming

Arrays, Vectors

Michael Conti

Department of Computer Science and Statistics
University of Rhode Island

Spring 2024

Question

- Write a program that reads in **3** values and outputs the same values in reverse order

- Write a program that reads in **n** values and outputs the same values in reverse order

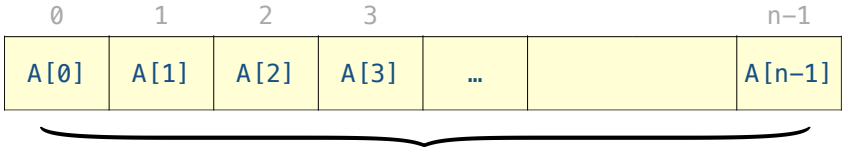
Arrays

Arrays

- An array is a **contiguous** sequence of elements of the **same type**

- Each element (data in array) can be accessed using its **index**

array name: A
array length: n



all elements of the same data type

Declaration

```
// array declaration by specifying size
int myarray1[100];

// can also declare an array of
// user specified size
int n = 8;
int myarray2[n];

// can declare and initialize elements
double arr[] = { 10.0, 20.0, 30.0, 40.0 };
// compiler figures the right size

// a different way
int arr[5] = { 1, 2, 3 };
// compiler creates an array of length 5 and
// initializes first 3 elements
```


Initialization and indexing

- Elements in an array **must be initialized** before use
  - otherwise, their initial values are **undetermined**
  - can use a loop to initialize values
- Individual elements can be accessed by using the **subscription operator [ ]**

```
int array[4];
array[0] = 5;
array[1] = array[0] + 10;
array[2] = array[1] + 20;
array[3] = array[2] + 30;
```

0	1	2	3
5	15	35	65

Trace the code

C++ (gcc 4.8, C++11)
EXPERIMENTAL! known bugs/limitations

```
1  int main() {
2    int val = 0;
3    int array[10];
4
5    for (int i = 0 ; i < 10 ; i++) {
6      val += 50;
7      array[i] = val;
8    }
9
10   return 0;
11 }
```

Stack Heap

main
val: int ?
array: array [0 1 2 3 4 5 6 7 8 9] int ? ? ? ? ? ? ? ? ? ?

C++ (gcc 4.8, C++11)
EXPERIMENTAL! known bugs/limitations

```
1  int main() {
2    int val = 0;
3    int array[10];
4
5    for (int i = 0 ; i < 10 ; i++) {
6      val += 50;
7      array[i] = val;
8    }
9
10   return 0;
11 }
```

Stack

main
val: int 500
array: array [0 1 2 3 4 5 6 7 8 9] int 50 100 150 200 250 300 350 400 450 500

http://pythontutor.com/cpp.html

Out of bounds?

- There is no **out of bounds** checking at compile time
  - **unexpected output**

A[9] ?

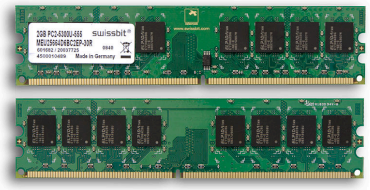
0	1	2	3	4	5	6	7				
?	?	?	10	20	50	100	70	50	30	5	?	?

What is the output?

```
#include <iostream>

int main() {
    int myarray[5];
    for (int i = 0 ; i < 5 ; i++) {
        myarray[i] = i;
    }
    for (int i = -10 ; i < 10 ; i++) {
        std::cout << myarray[i] << ' ';
    }
    std::cout << '\n';
    return 0;
}
```


Computer memory



- A **memory address** is a reference to a specific memory location
- Memory addresses are **fixed-length** sequences of digits (hexadecimal codes)
- **Word**-oriented memory organization (word size 32-bit in this illustration)

0x00000000	
0x00000004	
0x00000008	
0x0000000C	
0x00000010	
0x00000014	
0x00000018	
…	
…	
…	
0xFFFFFFEC	
0xFFFFFFF0	
0xFFFFFFF4	
0xFFFFFFF8	
0xFFFFFFFC	

address content

https://en.wikipedia.org/wiki/Random-access_memory

Computer memory (example)

```
int main() {
    int a = 4;
    int i = 0;
    double b = 10;
    int arr[5];

    for (; i < 5 ; i++) {
        arr[i] = i * 100;
    }

    return 0;
}
```

Assuming 32-bit words

…	
0x91340A04	
0x91340A08	4
0x91340A0C	5
0x91340A10	10
0x91340A14	
0x91340A18	0
0x91340A1C	100
0x91340A20	200
0x91340A24	300
0x91340A28	400
0x91340A2C	
0x91340A30	
0x91340A34	
…	

Passing arrays to functions

- When specifying the parameter, use **empty brackets**
- When providing the argument, use the **array name**
  - need to pass the **array length** separately

```
void zeros(int a[], int n) {
    for (int i = 0 ; i < n ; i ++) {
        a[i] = 0;
    }
}

int main() {
    int array[5];
    zeros(array, 5);
    // do stuff
}
```

Base address

- **Base address** is the memory location of the first element in an array
  - base address of `arr` is `0x91340A18` (previous example)
- When passing arrays to functions, the base address of the array is passed to the formal parameter

…	
0x91340A04	
0x91340A08	4
0x91340A0C	5
0x91340A10	10
0x91340A14	
0x91340A18	0
0x91340A1C	100
0x91340A20	200
0x91340A24	300
0x91340A28	400
0x91340A2C	
0x91340A30	
0x91340A34	
…	

Base address

Users > michaelconti > Desktop > arr.cpp

```
2
3   int main(){
4
5       int array[4];
6       array[0] = 5;
7       array[1] = array[0] + 10;
8       array[2]  = array[1] + 20;
9       array[3] = array[2] + 30;
10
11      std::cout << array;
12
13      return 0;
14  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

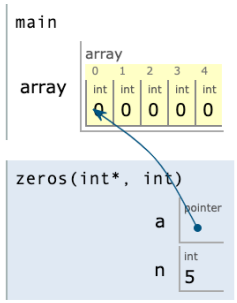
```
michaelconti@Michaels-MacBook-Pro-2 Desktop % ./temp
0x16ef638a8
michaelconti@Michaels-MacBook-Pro-2 Desktop %
```


Passing arrays to functions

C++ (gcc 4.8, C++11)
EXPERIMENTAL! known bugs/limitations

```
1  void zeros(int a[], int n) {
2      for (int i = 0 ; i < n ; i ++) {
3          a[i] = 0;
4      }
5  }
6
7  int main() {
8      int array[5];
9      zeros(array, 5);
10     // do stuff
11 }
```

Stack



http://pythontutor.com/cpp.html

Vectors

Vectors

- Data structor for organizing elements
- #include <vector>

```
// declare
std::vector<int> myVector;


// initializer list (c++17)
std::vector<int> vector1 = {1, 2, 3, 4, 5};
```


Declaration

```
// declare
std::vector<int> myVector;


// initializer list (c++17)
std::vector<int> vector1 = {1, 2, 3, 4, 5};
```


Important Methods

```
//declare vector without size
std::vector<int> myVector;

//declare vector with size
std::vector<int> myVector(20);

//add element into vector
myVector.push_back(5);

//add element into vector
myVector[0] = 5;

//access vector (with bound checking)
myVector.at(0);

//access vector (without bound checking)
myVector[0];

//change vector element
myVector[0] = 10;

//remove element into vector
myVector.pop_back();
```


Question

- Write a function that receives an array of integers and reverses the contents of the array

Question

- Write a function that receives an array and returns the smallest element in that array.