CS201 Introduction to Java Programming

Arrays

Chengyu Sun
California State University, Los Angeles

Collections of Data

- Student records
- Grocery lists
- Texts
- …

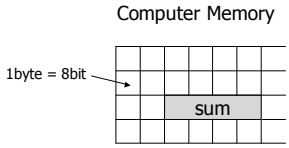
- A collection of elements with the same or similar type

Data Structures for Collections

- Lists, queues, stacks, trees … (topics of CS203 and CS312)
- Arrays
  - Simplest
  - Efficient in terms of memory and element access
  - Basis for the implementation of many other data structures
  - Fixed-size

Variables

- Name
- Type
- Value



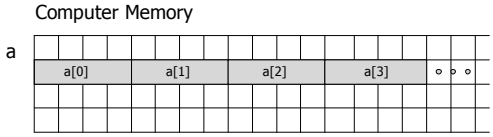

```
// declaration
int sum;

// assignment
sum = 0
```

// declaration and assignment
int sum=0;

Arrays

- Name
- Type
- Length (or Size) – number of elements in the array
- Values



Declaration

- Specify name and type of the array

```
// declare arrays a, b, c, d

int     a[];
char    b[], c[];
float[] d;

int     e[10]; // error! this is not C/C++
```

Allocation and Initialization

- Allocation – specify array length
- Initialization – specify element values

```
// declare an array a
int a[];

// allocate space for 10 integers.
// initial element values are 0 by default
a = new int[10];

// everything in one shot
char b[] = { 'C', 'h', 'e', 'n', 'g', 'y', 'u' };
```

Access Array Elements

- arrayname.length
- Index is from 0 to (arrayname.length-1)

```
int a[];
a = new int[10];

a[5] = 3;    // assign 3 to the 6th element

// prints out all elements
for( int i=0 ; i < a.length ; ++i )
  System.out.println( a[i] );
```

index

Array Example

- Find the index of the smallest element

```
int a[] = { 3, 28, 13, 2, 17, 1, 0 };

int index = 0;
for( int i=1 ; i < a.length ; ++i )
  if( a[i] < a[index] ) index = i;

System.out.println(index);
```

- Exercise: find the value of the smallest element

Multidimensional Data

Grades for CS201

	HW0	HW1	HW2
Student1	10	80	100
Student2	10	75	85
Student3	10	90	70
Student4	10	50	80

Image of a Rectangle

```
000000000000
011111111110
010000000010
010000000010
011111111110
000000000000
```

Multidimensional Array

```
int grades1[][];

grades1 = new int[??][??];

int grades2 = { {10, 80, 100}, {10, 75, 85},
                {10, 90, 70}, {10, 50, 80} };

/** element access */

for( int i=0 ; i < ?? ; ++i )  // for each row
  for( int j=0 ; j < ?? ; ++j )  // for each column
    System.out.println( grades2[?][?] );
```

Multidimensional Array as Array-of-Arrays

```
int grades2 = { {10, 80, 100}, {10, 75, 85},
                {10, 90, 70}, {10, 50, 80} };
```

- grades2 – an array of 4 arrays
  - grades2[0] – an array of 3 integers: `10`, `80`, and `100`
  - …
  - grades2[3] – an array of 3 integers: `10`, `50`, and `80`

arrayname.length and Friends

- arrayname.length is the length of the 1st dimension
- arrayname[i].length is the length of the 2nd dimension
- arrayname[i][j].length is the length of the 3rd dimension
- …

More Fun with Multidimensional Array

- Each row doesn't have to have the same number of elements

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
int a[][];

// allocation
a = new int[4][];
for( int i=0 ; i < a.length ; ++i )
  a[i] = new int[??];
```

- Exercise: add initialization to the code above

Array Parameter

- Write a method sumA() which returns the sum of the elements in a given array

```
int sumA( int a[] )
{
   int sum = 0;

   ??

   return sum;
}
```

Parameter Passing Example

```
…
int a = 10, b = 20;
swap1( a, b );

system.out.println(a);
system.out.println(b);
…

void swap1( int x, int y)
{
   int tmp = x;
   x = y;
   y = tmp;
}
```

```
…
int a[] = { 10, 20 };
swap2( a );

system.out.println(a[??]);
system.out.println(a[??]);
…

void swap2( int x[])
{
   int tmp = x[??];
   x[??] = x[??];
   x[??] = tmp;
}
```

Parameter Passing

- Pass by value
  - All primitive types
  - Safe
  - May not be efficient
- Pass by reference
  - All class types, including arrays
  - Less safe
  - Efficient

Array and Recursion

- Write a recursive method to return the smallest number in an array
  - `int min( int a[] )`
  - The recursive *helper* method
    - `int recurMin( int a[], int index )`

The Search Problem

- Given an array and a value, find the index of the array element which has the given value
  - Index = -1 if the value is not in the array
  - For simplicity, assume all elements have distinguish values

{3, 28, 13, 2, 17, 1, 0}

28 → 1
17 → 4
128 → -1

Sequential Search

- Simple, but slow
- Number of element accessed
  - Best case: 1
  - Worse case: N
  - Average case: N/2

```
int index = -1; // not found yet
for( int i=0 ; i < a.length ; ++i )
{
   if( a[i] == value )
   {
      index = i; // found
      break; // so we stop here
   }
}
```

Binary Search

Search for 28

{ 0, 1, 2, 3, 13, 17, 28 }
{ 0, 1, 2, 3, 13, 17, 28 }
{ 0, 1, 2, 3, 13, 17, 28 }

Search for 15

{ 0, 1, 2, 3, 13, 17, 28 }
{ 0, 1, 2, 3, 13, 17, 28 }
{ 0, 1, 2, 3, 13, 17, 28 }

Binary Search – Code

```
// assume a[] is sorted in ascending order

int index = -1;
int left = 0, right = a.length, mid;

while( ?? )
{
   mid = (left+right)/2;
   if( a[mid] > value ) right = mid-1;
   else if( a[mid] < value ) left = mid+1;
   else
   {
      index = mid;
      break;
   }
}
```

Bubble Sort

- Find a smallest element
- Put it in the 1st position
- Find the 2nd smallest element
- Put it in the 2nd position
- ...

{ 3, 28, 13, 2, 17, 1, 0 }
{ 0, 28, 13, 2, 17, 1, 3 }
{ 0, 1, 13, 2, 17, 28, 3 }
{ 0, 1, 2, 13, 17, 28, 3 }
{ 0, 1, 2, 3, 17, 28, 13 }
{ 0, 1, 2, 3, 13, 28, 17 }
{ 0, 1, 2, 3, 13, 17, 28 }

Bubble Sort – Code

```
// sort a[] into ascending order

int left = 0;
while( ?? )
{
   int index = left;
   for( int i=left ; i < a.length ; ++i )
      if( a[i] < a[index] ) index = i;

   // swap a[index] and a[left]
   int tmp = a[index];
   a[index] = a[left];
   a[left] = tmp;

   ??
}
```