C Programming: Arrays

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

Aug 28, 2017

Outline

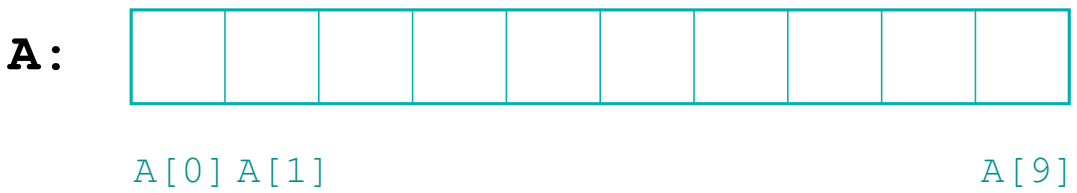
Array declaration and access

An array is a contiguous set of cells in memory, each of which can hold an object of the declared type.

Example: the declaration

```
int A[10];
```

declares an array of 10 `int`s, accessed as `A[0]`, ..., `A[9]`.



We can assign to array elements as follows:

```
A[0] = 31;
```

`A[exp]` can be used wherever a variable (of the same type) could be used.

Day-of-Year example using arrays

```
void main() {
  int d, m, doy, i;
  int mdays[12];

  mdays[0] = 31; // Jan
  mdays[1] = 28; // Feb
  mdays[2] = 31;
  mdays[3] = 30;
  mdays[4] = 31;
  mdays[5] = 30;
  ...
  mdays[10] = 30;
  mdays[11] = 31; // Dec

  printf("Enter day (1-31) and month (1-12)\n");
  scanf("%d %d", &d, &m);
  doy = 0;
  for (i = 0; i < (m - 1); i++)
    doy += mdays[i];
  doy += d;
  printf("Day of 2017 is: %d\n", doy);
}
```

Exercise

Exercise

Write a main function that

- declares an array of 20 `ints`
- initializes it to 0, 10, 20, ..., 190,
- and finally prints the contents of the array.

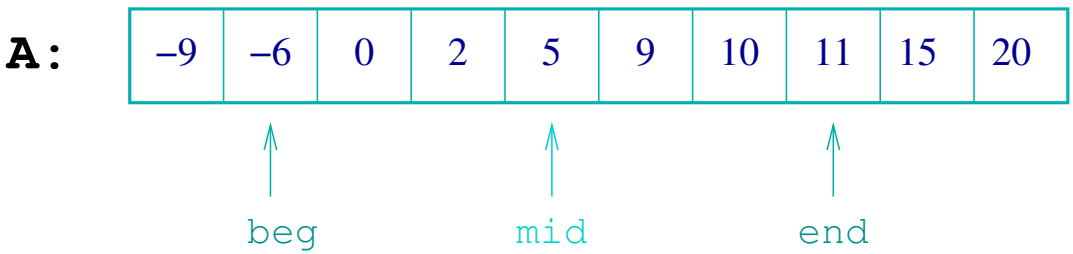
Linear search in an array

```
#define SIZE 1000000

void main() {
  int A[SIZE];
  int pos;
  pos = rand() % SIZE;
  A[pos] = 17;
  pos = linsearch(A, SIZE, 17);
  if (pos > -1)
    printf("17 occurs at position %d\n", pos);
  else
    printf("17 does not occur\n");
}

int linsearch(int a[], int size, int val) {
  int i;
  for (i = 0; i < size; i++)
    if (a[i] == val)
      return i;
  return -1;
}
```

Binary search in an array



Binary search implementation

```
void main() {
  int A[SIZE]; int pos;
  // initialize A to sorted ascending order ...
  pos = bsearch(A, 0, SIZE-1, 17);
  if (pos > -1)
    printf("11 occurs at position %d\n", pos);
  else
    printf("11 does not occur\n");
}

int bsearch(int a[], unsigned int beg, unsigned int end, int val) {
  int mid;
  if (beg == end)
    if (a[beg] == val)
      return beg;
    else
      return -1;
  mid = (beg + end) / 2;
  if (val <= a[mid])
    return bsearch(a, beg, mid, val);
  else
    return bsearch(a, mid+1, end, val);
}
```

Discussion on running time

- How many steps does linear search take?
- How many steps does binary search take?
- What would be the effect on running time on large arrays, with complex comparison operations?

Recursive functions

- A function (like `bsearch`) is called recursive if it (directly or indirectly) calls itself.
- Recursive functions are nothing special. Execution takes place as for normal functions: with each function invocation a new "frame" of local variables is created.
- However, need to be careful to avoid non-termination.

Recursive functions and termination: factorial example

```
int fact(int n) {
  if (n <= 1)
    return 1;
  else
    return n * fact(n-1);
}
```

```
int fact(int n) {
  return n * fact(n-1);
}
```