Arrays, Sorting in C
M. Tech. (CS) 1st year, 2015

Arijit Bishnu
arijit@isical.ac.in

Indian Statistical Institute, India.

August 4, 2015

Outline

1. C as an imperative language
2. Arrays in C
3. Finding the maximum and minimum
4. Sorting

Outline

1. C as an imperative language
2. Arrays in C
3. Finding the maximum and minimum
4. Sorting

A brief introduction

- C is an imperative language.

A brief introduction

- C is an imperative language.
- Imperative programs consist of

A brief introduction

- C is an imperative language.
- Imperative programs consist of
  - a program state – how is it encoded?

A brief introduction

- C is an imperative language.
- Imperative programs consist of
  - a program state – how is it encoded?
  - instructions/commands that change the program state – what are they? (assignment statements, conditionals, loop, procedures)

A brief introduction

- C is an imperative language.
- Imperative programs consist of
  - a program state – how is it encoded?
  - instructions/commands that change the program state – what are they? (assignment statements, conditionals, loop, procedures)
- Instructions in an imperative language are similar to the native machine instructions of traditional computer hardware.

A brief introduction

- C is an imperative language.
- Imperative programs consist of
  - a program state – how is it encoded?
  - instructions/commands that change the program state – what are they? (assignment statements, conditionals, loop, procedures)
- Instructions in an imperative language are similar to the native machine instructions of traditional computer hardware.
- A bit of history – John von Neumann's *stored program computers* and Eckert & Mauchly's contribution. The main idea is that machine can store both data and instructions indistinguishably.

Outline

1. C as an imperative language
2. Arrays in C
3. Finding the maximum and minimum
4. Sorting

Arrays in C

- Declaring an array — (`data type`) (`array_name[size]`)
  Example: `unsigned int  loc[100];`

Arrays in C

- Declaring an array — (`data type`) (`array_name[size]`)
  Example: `unsigned int  loc[100];`
- Accessing array elements — array indexing starts from 0 (zero);
  Example: `loc[0]`, ..., `loc[i]`, ..., `loc[100]` (Can you notice an error?)
  Message: Beware of going out of range!

Arrays in C

- Declaring an array — (`data type`) (`array_name[size]`)
  Example: `unsigned int  loc[100];`
- Accessing array elements — array indexing starts from 0 (zero);
  Example: `loc[0]`, ..., `loc[i]`, ..., `loc[100]` (Can you notice an error?)
  Message: Beware of going out of range!
- Arrays and strings — Let's see an example program.

Outline

1 C as an imperative language

2 Arrays in C

3 Finding the maximum and minimum

4 Sorting

Order Statistics

- Let us find the maximum (minimum) in an array of n elements;

Order Statistics

- Let us find the maximum (minimum) in an array of n elements;
- Let us count the number of comparisons; It is $n - 1$

Order Statistics

- Let us find the maximum (minimum) in an array of n elements;
- Let us count the number of comparisons; It is $n - 1$
- Let us find the second maximum; what is the number of comparisons?

Order Statistics

- Let us find the maximum (minimum) in an array of n elements;
- Let us count the number of comparisons; It is $n - 1$
- Let us find the second maximum; what is the number of comparisons?
- Let us find the maximum and minimum together; what is the number of comparisons?

Exercises

Second maximum

Find the second maximum in an array of n elements efficiently. Trivially, you can do it in $2n - 3$ comparisons.

Maximum and minimum

Find the maximum and minimum in an array of n elements efficiently. Trivially, you can do it in $2n - 2$ comparisons.

Outline

1. C as an imperative language
2. Arrays in C
3. Finding the maximum and minimum
4. Sorting

Sorting

- Bubble sort – Look at consecutive elements of the array $\mathcal{A}$ and push the maximum to the right. This way the maximum bubbles up to the last location, i.e., $(n-1)$-th position of $\mathcal{A}$. We now do the same starting from the first location again so that the second maximum bubbles up to the $(n-2)$-th position of $\mathcal{A}$. This continues. This takes $O(n^2)$ comparisons. A model code is given.

Sorting

- Bubble sort – Look at consecutive elements of the array $\mathcal{A}$ and push the maximum to the right. This way the maximum bubbles up to the last location, i.e., $(n-1)$-th position of $\mathcal{A}$. We now do the same starting from the first location again so that the second maximum bubbles up to the $(n-2)$-th position of $\mathcal{A}$. This continues. This takes $O(n^2)$ comparisons. A model code is given.
- Selection sort – The maximum element in $\mathcal{A}$ and its location, say j, is found; the last element of $\mathcal{A}$ is swapped with the j-th location, so the last location of $\mathcal{A}$ has the maximum element. Keep doing this on the rest of the array apart from the maximum which is at the last location now. This takes $O(n^2)$ comparisons. A model code is given.