Data structures and algorithms

Tutorial 2

Amr Keleg

Faculty of Engineering, Ain Shams University

March 5, 2019

Contact: amr_mohamed@live.com

Outline

Outline

Given an array, implement a function to sort it.

```
void sort(int arr[], int arr_len){
  // Apply a sorting algorithm
}
```

Outline

As long as the array isn't sorted:
Find two successive elements such that arr[i] is bigger than arr[i+1], then swap them.

Outline

```
for (int i=0; i<arr_len -1; i++){
  if (arr[i] > arr[i+1])
    swap(arr[i], arr[i+1]);
}
```

What is the required number of iterations to ensure that the array is sorted?

HINT: What happens to the array after one iteration?

Outline

```
void sort(int arr[], int arr_len){
  for (int iter=0; iter< arr_len; iter++){
    for (int i=0; i<arr_len - 1; i++){
      if (arr[i] > arr[i+1])
        swap(arr[i], arr[i+1]);
    }
  }
}
```

What is the complexity of the bubble sort?

Outline

```
void sort(int arr[], int arr_len){
  for (int iter=0; iter< arr_len -1; iter++){
    bool swapped = false;
    for (int i=0; i<arr_len -1-iter; i++){
      if (arr[i] > arr[i+1]){
        swap(arr[i], arr[i+1]);
        swapped = true;
      }
    }
    if (!swapped){
      return ;
    }
  }
}
```

What is the complexity of improved bubble sort?

Visualization: `https://visualgo.net/bn/sorting`

Outline

Outline

```
int arr_len = 5;
int arr[arr_len];
for (int indx=0; indx< arr_len; indx++){
  cin>>arr[indx];
}
```

What is complexity of accessing an element in the array?

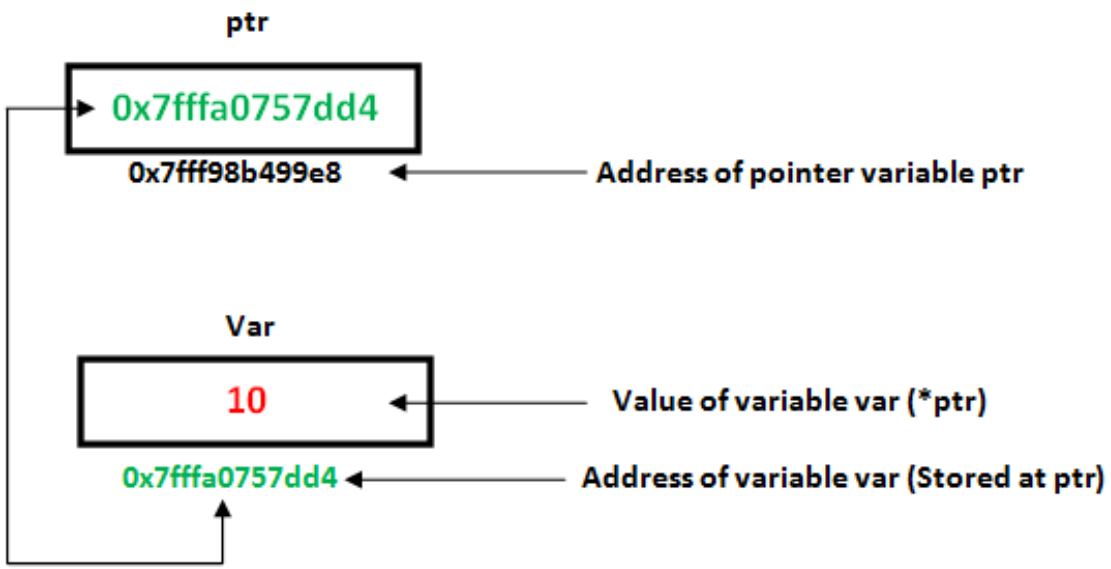
Outline

```
int var = 20;

//declare pointer variable
int *ptr;

//note that data type of ptr and var must be same
ptr = &var;

// assign the address of a variable to a pointer
cout << ptr << "\n"; // Address of var in memory
cout << var << "\n"; // 20
cout << *ptr << "\n"; // 20
```


ptr
0x7fffa0757dd4
0x7fff98b499e8
Address of pointer variable ptr
Var
10
Value of variable var (*ptr)
0x7fffa0757dd4
Address of variable var (Stored at ptr)

Outline

```
int arr_len = 5;
int arr[arr_len];
for (int indx=0; indx< arr_len; indx++){
  cin>>arr[indx];
}

// Address of the first element of arr in memory
cout << arr <<"\n";
// Value of arr[0]
cout << *arr << "\n";
// Address of the second element of arr in memory
cout << arr+1 << "\n";
// Value of arr[1]
cout << *(arr+1) << "\n";
```

Outline

```
int * arr;
int arr_len;
cin>>arr_len;
arr = new int[arr_len];
for (int indx=0; indx< arr_len; indx++){
   cin>>arr[indx];
}
```

Outline

Create a dynamic array of float numbers, the size of the array is determined by the user through cin, each element in the array holds a value of 1/(index)! i.e a[i]=1.0/i!, run your program and compute the sum of the array elements (which value the sum tends to ?)

```
float * arr;
int arr_len;
cin>>arr_len;
arr = new float[arr_len];
float fact_i = 1;
arr[0] = 1;
for (int i=1;i<arr_len;i++){
   fact_i *= i;
   arr[i] = 1.0 / fact_i;
}
float sum = 0;
for (int i=0; indx< arr_len; i++){
  sum += arr[i];
}
cout << sum;
```

Outline

Outline

Find the complexity of an algorithm whose running time is:

$$T(n) = 2 * T(n/2) + n^2 \ \textit{if} \ \ n > 1$$

$$T(1) = 1$$

$$\textit{Complexity is } O(n^2)$$

Find the complexity of an algorithm whose running time is:

$$T(n) = 8 * T(n/2) + 1000n^2 \ \textit{if} \ \ n > 1$$

$$T(1) = 1000$$

$$\textit{Complexity is } O(n^3)$$

Find the complexity of an algorithm whose running time is:

$$T(n) = 2 * T(n/2) + 10 * n \ \textit{if} \ \ n > 1$$

$$T(1) = 1$$

$$\textit{Complexity is } O(nlog(n))$$

Outline

Outline

```
class email_book{
  private:
    int MAX_SIZE;
    string * emails;
    string * names;
    int book_size;
  public:
    email_book(int MAX_SIZE=100){
      this->MAX_SIZE = MAX_SIZE;
      emails = new string[MAX_SIZE];
      names = new string[MAX_SIZE];
      book_size = 0;
    }
    string get_name(int index){
      return names[index];
    }
    string set_name(int index, string name){
      return names[index] = name;
    }
};
```

Outline

The fibonacci sequence is defined as:

$$F[0] = 0$$

$$F[1] = 1$$

$$F[n] = F[n-1] + F[n-2] \ for \ n > 2$$

Write a recursive function to compute the fibonacci sequence at index n.

```
long long int fib(int n);
```

```
long long int fib(int n){
  // Base cases
  if (n < 2)
    return n;

  // Recursive calls
  return fib(n-1) + fib(n-2);
}
```

Feedback form: `https://goo.gl/forms/Ipl1WlDVv8gCwn1a2`