Data structures and algorithms

Tutorial 2

Amr Keleg

Faculty of Engineering, Ain Shams University

March 3, 2020

Contact: `amr_mohamed@live.com`

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++

3 Example on a Class in C++

4 Our second data structure - Linked List

5 References and feedback

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

```
const 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

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

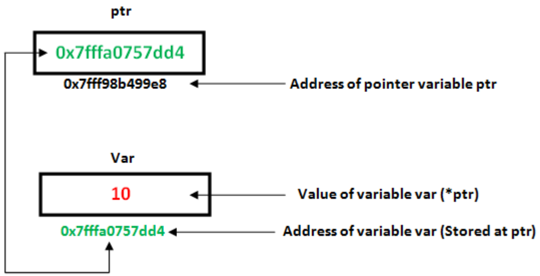
4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

```
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

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

```
const 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

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

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

// DON'T FORGET TO DELETE A DYNAMICALLY ALLOCATED ARRAY
delete [] arr;
```

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

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; i< arr_len; i++){
  sum += arr[i];
}
cout << sum;

delete [] arr;
```

Try other data-types for the variable fact_i
Is fact_i a decimal (floating point) value?
Print the values of fact_i if arr_len was 30, Is there anything strange?
What does the sum tend to?

Outline

1 Our first data-structure (Arrays)

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++

4 Our second data structure - Linked List

5 References and feedback

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

Will this code compile and run?

```
#include<iostream>
using namespace std;

int main(){
  int size;
  cin>>size;
  int arr[size];
}
```

Will this code compile and run?

```
#include<iostream>
using namespace std;

int main(){
  int size;
  cin>>size;
  int arr[size];
}
```

- Yes, If you are using codeblocks (g++/gcc compilers)
- No, If you are using Visual Studio

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

VLA (Variable Length Arrays)

- Allows the creation of arrays in the stack whose size is determined at run-time.
- treated like local variables. The compiler deletes them when the function terminates.
- Not part of the standard C++ specification.

VLA (Variable Length Arrays)

- g++/gcc have extensions that allow the usage of VLA.
- g++ vla.cpp -o vla.exe
- Disable the extension and try again: g++ -Wvla vla.cpp -o vla.exe
- The Linux Kernel Is Now VLA-Free: A Win For Security, Less Overhead & Better For Clang.
- `https://www.phoronix.com/scan.php?page=news_item&px=Linux-Kills-The-VLA`

Outline

1 Our first data-structure (Arrays)

2 Demystifying Arrays creation in C++

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List

5 References and feedback

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

Create a class for email address book. Your class should have the following:

- An array of strings containing email followed by the person name, an integer number to tell how many emails are stored so far. (private)
- gets and sets methods for email and name for each entry. (public)
- Print_all method to list all entry values. (public OR private?)
- a constructor method to initialize the size used so far to zero. (public OR private?)

```
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;
    }

    void print_all(){
      for(int i=0; i< book_size; i++)
        cout<<"The email of "<<names[i]<<" is: "<<emails[i]<<endl;
    }

    ~email_book(){
      delete [] emails;
      delete [] names;
    }
};
```

Outline

1 Our first data-structure (Arrays)

2 Demystifying Arrays creation in C++

3 Example on a Class in C++

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

5 References and feedback

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

How to insert an element at the beginning of an array?

```
void insert_at_beginning(int element, int arr[],
int arr_len, int max_arr_len){

  // INSERT the element

}
```

```c
void insert_at_beginning(int element, int arr[],
int arr_len, int max_arr_len){

  for (int index=arr_len-1; index>=0; index--){
    arr[index + 1] = arr[index];
  }

  arr[0] = element;
}
```

The complexity is O(n).
What happens if the array has reached its maximum size?
1. Create a new array of bigger size.
2. Copy the data from the old array to the new one.
3. Add the new element at the beginning of the array.
(The same applies for appending an element to the end of the array).

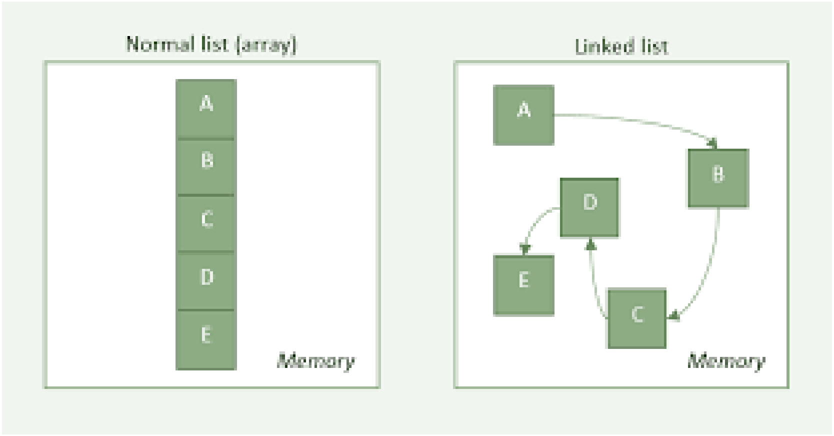
Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class


Normal list (array)
A
B
C
D
E
Memory
Linked list
A
B
D
E
C
Memory

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

```
class Node{
  // carries data and pointer to the following element
};
```

```
class Node{
  private:
    int data;
    Node * next;

  public:
    Node(int d){
      this->data = d;
      this->next = nullptr;
    }

    void set_next(Node * next){
      this->next = next;
    }
    Node * get_next(){
      return next;
    }
    // Same for data
    friend class LinkedList;
};
```

Outline

1 Our first data-structure (Arrays)
- Basic C++ syntax
- Quick introduction to pointers
- Array name as a pointer
- Basic C++ syntax (Dynamic Array)
- Sheet 1 - Question 5

2 Demystifying Arrays creation in C++
- Will this code snippet compile?
- VLA (Variable Length Arrays)

3 Example on a Class in C++
- Sheet 1 - Question 6

4 Our second data structure - Linked List
- Why do we need it?
- The linked list
- Implementation details - Node class
- Implementation details - LinkedList class

```
class LinkedList{
  private:
    Node * head;

  public:
    LinkedList();
    bool empty();
    int length();
    void push_front(int d);
    void pop_front();
    void push_back(int d);
    void pop_back();
    void print();
    bool contains(int d);
    void clear();
    void bubble_sort();
    ~LinkedList();
};
```

```
LinkedList(){
    head = nullptr;
}
```

```
bool empty(){
  return head == nullptr;
}
```

```
void push_front(int d){
  Node * new_head = new Node(d);
  new_head->next = head;
  head = new_head;
}
```

```
int length(){
  Node * cur_node = head;
  int length = 0;
  while(cur_node != nullptr){
    length++;
    cur_node = cur_node->next;
  }

  return length;
}
```

```
void pop_front(){
  if (this->empty()){
    return;
  }
  Node * head_to_be_deleted;
  head_to_be_deleted = head;
  head = head->next;
  delete head_to_be_deleted;
}
```

- Visualizations of linked list: `https://visualgo.net/bn/list`
- Really interesting exercises on linked list: `https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=linked-lists`

Outline

1 Our first data-structure (Arrays)

2 Demystifying Arrays creation in C++

3 Example on a Class in C++

4 Our second data structure - Linked List

5 References and feedback

STL (Standard Template Library):

- Vectors:
  https://syntaxdb.com/ref/cpp/vectors
  List of functions:
  http://www.cplusplus.com/reference/vector/vector/
- Lists:
  http://www.cplusplus.com/reference/list/list/

Reference:
Data Structure and Algorithm Analysis in C++, 3rd edition, Mark Allen Weiss.

- 1.5 C++ Details
- 1.6 Template

Feedback form:
Amr: https://forms.gle/3VVidgqk6mszbpBt8
Fady: https://forms.gle/NjK6UbgiGD16ZWYF6