Advanced Programming in C++

فرشاد حکیم پور

Exercise

- Implement the queue by a linked list
- Implement the Bubble Sort algorithm for an array of k integer numbers
- Implement Bubble Sort for a linked list
  - Send the c++ file to **tamrin.ut@gmail.com**
  - Add EX2 - <Your Stud. No.> to the subject line
  - Deadline: 13th - Azar

Memory Allocation

- Dynamic versus Static memory allocation
- Amount of *static* memory required by the program is known at the compilation time
- Memory requested and allocated at the run time is known as *dynamic* memory

Static Memory Allocation

```
Node *start;
Node *end;
Node n = Node('c');
start = &n;
end = &n;
Node m = Node('+');
end->setNext(m);
end = &m;
```


Dynamic Memory Allocation

```
n = new Node('c');
start = n;
end = n;

n = new Node('+');
end->setNext(n);
end = n;

n = new Node('+');
end->setNext(n);
end = n;

cout << start->getValue();
cout << start->getNext()->getValue();
cout << start->getNext()->getNext()->getValue() << endl;
```


Suggestion

- Before you implement the queue:
  - Implement a linklist object
  - Consider a cursur to work with your list
  - Possible methods:
    - insert(node n)
    - Node *delete()
    - Node *get()
    - void first()
    - bool last()
    - void next()

"const" Keyword

- "const" keyword indicates that you do not intend to change the value of the variable in your code
- It is called *named constant*, read-only variable or constant variable

```
const int arraysize = 10;
int varArray [arraysize];
arraysize = 15;
```

Compilation Error

Variable Length Array

- Amount of memory consumed by a variable length array varies during time
- The following code *should* cause a compilation error (it may not -if not it is not a good coding style):

```
int i;
std::cin >> i;
int varLenArray [i];
```


Variable Length Array

- In Standard C++ code use the vector class (from Standard Tamplate Library)

```
#include <vector>
using std::vector;
Vactor<int> v;
```

- Vectors have more features compared to arrays
- "v.size()" returns the size of the array
- "v.at(i)" returns the element at the i^{th} position controlling the boundary of the array (unlike[])