

# Δομές Δεδομένων και Αλγόριθμοι - Εργαστήριο 5

## Στοιίβες (stacks) και Ουρές (queues), οι δομές στοίβα και ουρά στην STL

Τ.Ε.Ι. Ηπείρου, Τμήμα Μηχανικών Πληροφορικής Τ.Ε.  
Χρήστος Γκόγκος - Αναπληρωτής Καθηγητής

### 1 Εισαγωγή

Οι στοιίβες και οι ουρές αποτελούν απλές δομές δεδομένων που είναι ιδιαίτερα χρήσιμες στην επίλυση αλγοριθμικών προβλημάτων. Η στοιίβα είναι μια λίστα στοιχείων στην οποία τα νέα στοιχεία τοποθετούνται στην κορυφή και όταν πρόκειται να αφαιρεθεί ένα στοιχείο αυτό πάλι συμβαίνει από την κορυφή των στοιχείων της στοιίβας. Από την άλλη μεριά η ουρά είναι επίσης μια λίστα στοιχείων στην οποία όμως οι εισαγωγές γίνονται στο πίσω άκρο της ουράς ενώ οι εξαγωγές πραγματοποιούνται από το εμπρός άκρο της ουράς. Στο εργαστήριο αυτό θα παρουσιαστούν υλοποιήσεις της στοιίβας και της ουράς. Επιπλέον, θα παρουσιαστούν οι δομές της STL `std::stack` και `std::queue`. Ο κώδικας όλων των παραδειγμάτων βρίσκεται στο [https://github.com/chgogos/ceteiep\\_dsa](https://github.com/chgogos/ceteiep_dsa).

### 2 Στοιίβα

Η στοιίβα (stack) είναι μια ειδική περίπτωση γραμμικής λίστας στην οποία οι εισαγωγές και οι διαγραφές επιτρέπονται μόνο από το ένα άκρο. Συνήθως αυτό το άκρο λέγεται κορυφή (top). Πρόκειται για μια δομή στην οποία οι εισαγωγές και οι εξαγωγές γίνονται σύμφωνα με το μοντέλο τελευταίο μέσα πρώτο έξω (LIFO=Last In First Out).

Στη συνέχεια παρουσιάζεται μια υλοποίηση στοιίβας που χρησιμοποιεί για την αποθήκευση των στοιχείων της έναν πίνακα. Εναλλακτικά, στη θέση του πίνακα μπορεί να χρησιμοποιηθεί συνδεδεμένη λίστα. Μια υλοποίηση στη γλώσσα C μπορεί να βρεθεί στην αναφορά [1].

```
1 #include <iostream>
2
3 using namespace std;
4
5 template <typename T> class my_stack {
6 private:
7     T *data;
8     int top, capacity;
9
10 public:
11     // constructor
12     my_stack(int c) {
13         top = -1;
14         capacity = c;
15         data = new T[capacity];
16     }
17
18     // destructor
19     ~my_stack() { delete[] data; }
20 }
```

```

21 bool empty() { return (top == -1); }
22
23 void push(T elem) {
24     if (top == (capacity - 1))
25         throw "The stack is full";
26     else {
27         top++;
28         data[top] = elem;
29     }
30 }
31
32 T pop() {
33     if (top == -1)
34         throw "the stack is empty";
35     top--;
36     return data[top + 1];
37 }
38
39 void print() {
40     for (int i = 0; i <= top; i++)
41         cout << data[i] << " ";
42     cout << endl;
43 }
44 };
45
46 int main() {
47     my_stack<char> astack(100);
48     for (char c = 65; c < 65 + 26; c++)
49         astack.push(c);
50     astack.print();
51     while (!astack.empty()) {
52         cout << astack.pop() << " ";
53     }
54     cout << endl;
55 }

```

Κώδικας 1: Υλοποίηση στοίβας (stack\_oo.cpp)

---

```

1 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
2 Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

```

---

### 3 Ουρά

Η ουρά (queue) είναι μια ειδική περίπτωση γραμμικής λίστας στην οποία επιτρέπονται εισαγωγές στο πίσω άκρο της και εξαγωγές από το εμπρός άκρο της μόνο. Τα δύο αυτά άκρα συνήθως αναφέρονται ως πίσω (rear) και εμπρός (front) αντίστοιχα. Η ουρά είναι μια δομή στην οποία οι εισαγωγές και οι εξαγωγές γίνονται σύμφωνα με το μοντέλο πρώτο μέσα πρώτο έξω (FIFO=First In First Out).

Στη συνέχεια παρουσιάζεται μια υλοποίηση ουράς στην οποία τα δεδομένα της τοποθετούνται σε έναν πίνακα (εναλλακτικά θα μπορούσε να είχε χρησιμοποιηθεί μια άλλη δομή όπως για παράδειγμα η συνδεδεμένη λίστα). Ο πίνακας λειτουργεί κυκλικά, δηλαδή όταν συμπληρωθεί και εφόσον υπάρχουν διαθέσιμες κενές θέσεις στην αρχή του πίνακα, τα νέα στοιχεία που πρόκειται να εισαχθούν στην ουρά τοποθετούνται εκεί.

---

```

1 #include <iostream>
2
3 using namespace std;
4
5 template <typename T> class my_queue {
6 private:

```

---

```

7  T *data;
8  int front, rear, capacity, size;
9
10 public:
11     // constructor
12     my_queue(int c) {
13         front = 0;
14         rear = -1;
15         size = 0;
16         capacity = c;
17         data = new T[capacity];
18     }
19
20     // destructor
21     ~my_queue() { delete[] data; }
22
23     bool empty() { return (size == 0); }
24
25     void enqueue(T elem) {
26         if (size == capacity)
27             throw "The queue is full";
28         else {
29             rear++;
30             rear %= capacity;
31             data[rear] = elem;
32             size++;
33         }
34     }
35
36     T dequeue() {
37         if (size == 0)
38             throw "the queue is empty";
39         T x = data[front];
40         front++;
41         front %= capacity;
42         size--;
43         return x;
44     }
45
46     void print(bool internal = true) {
47         for (int i = front; i < front + size; i++)
48             cout << data[i % capacity] << " ";
49         cout << endl;
50         if (internal)
51             for (int i = 0; i < capacity; i++)
52                 if (front <= rear && i >= front && i <= rear)
53                     cout << "[" << i << "] ->" << data[i] << " ";
54                 else if (front >= rear && (i >= front || i <= rear))
55                     cout << "[" << i << "] ->" << data[i] << " ";
56                 else
57                     cout << "[" << i << "] ->X ";
58         cout << " (front:" << front << " rear:" << rear << ")" << endl;
59     }
60 };
61
62 int main() {
63     my_queue<int> aqueue(10);
64     cout << "1. Enqueue 10 items" << endl;
65     for (int i = 51; i <= 60; i++)
66         aqueue.enqueue(i);
67     aqueue.print();

```

```

68 cout << "2. Dequeue 5 items" << endl;
69 for (int i = 0; i < 5; i++)
70     aqueue.dequeue();
71 aqueue.print();
72 cout << "3. Enqueue 3 items" << endl;
73 for (int i = 61; i <= 63; i++)
74     aqueue.enqueue(i);
75 aqueue.print();
76 }

```

Κώδικας 2: Υλοποίηση ουράς (queue\_oo.cpp)

```

1 1. Enqueue 10 items
2 51 52 53 54 55 56 57 58 59 60
3 [0]→51 [1]→52 [2]→53 [3]→54 [4]→55 [5]→56 [6]→57 [7]→58 [8]→59 [9]→60 (front:0 rear:9)
4 2. Dequeue 5 items
5 56 57 58 59 60
6 [0]→X [1]→X [2]→X [3]→X [4]→X [5]→56 [6]→57 [7]→58 [8]→59 [9]→60 (front:5 rear:9)
7 3. Enqueue 3 items
8 56 57 58 59 60 61 62 63
9 [0]→61 [1]→62 [2]→63 [3]→X [4]→X [5]→56 [6]→57 [7]→58 [8]→59 [9]→60 (front:5 rear:2)

```

## 4 Οι δομές στοίβα και ουρά στην STL

Οι δομές `std::stack` και `std::queue` έχουν υλοποιηθεί στην STL ως container adaptors δηλαδή κλάσεις που χρησιμοποιούν εσωτερικά ένα άλλο container και παρέχουν ένα συγκεκριμένο σύνολο από λειτουργίες που επιτρέπουν την προσπέλαση και την τροποποίηση των στοιχείων τους. Το εσωτερικό container μπορεί να είναι κάποιο από τα containers της STL: `vector`, `list`, `deque` ή οποιοδήποτε container που υποστηρίζει τις λειτουργίες: `empty`, `size`, `back`, `push_back` και `pop_back`.

### 4.1 `std::stack`

Τυπικές λειτουργίες που παρέχει η `std::stack` είναι οι ακόλουθες:

- `empty`, ελέγχει αν η στοίβα είναι άδεια.
- `size`, επιστρέφει το μέγεθος της στοίβας.
- `top`, προσπελαύνει το στοιχείο που βρίσκεται στη κορυφή της στοίβας (χωρίς να το αφαιρεί).
- `push`, ωθεί ένα στοιχείο στη κορυφή της στοίβας
- `pop`, αφαιρεί το στοιχείο που βρίσκεται στη κορυφή της στοίβας.

Ένα παράδειγμα χρήσης της `std::stack` παρουσιάζεται στη συνέχεια.

```

1 #include <deque>
2 #include <iostream>
3 #include <list>
4 #include <stack>
5 #include <vector>
6
7 using namespace std;
8 int main(void) {
9     cout << "std::stack example" << endl;
10    stack<char> items; // adapter over a vector container
11    // stack<char, vector<char>> items; // adapter over a vector container
12    // stack<char, list<char>> items; // adapter over a list container
13    // stack<char, deque<char>> items; // adapter over a deque container
14
15    for (char c = 65; c < 65 + 26; c++) {
16        cout << c << " ";

```

```

17     items.push(c);
18 }
19 cout << endl;
20
21 while (!items.empty()) {
22     cout << items.top() << " ";
23     items.pop();
24 }
25 cout << endl;
26 }

```

Κώδικας 3: Παράδειγμα χρήσης της `std::stack` (`stl_stack_example.cpp`)

```

1 $ stl_stack_example
2 Usage examples:
3     stl_stack_example SOFOS
4     stl_stack_example "A MAN A PLAN A CANAL PANAMA"
5
6 $ stl_stack_example "A MAN A PLAN A CANAL PANAMA"
7 The sentence A MAN A PLAN A CANAL PANAMA is palindromic.
8
9 $ stl_stack_example SOFIA
10 The sentence SOFIA is not palindromic.

```

## 4.2 `std::queue`

Τυπικές λειτουργίες που παρέχει η `std::queue` είναι οι ακόλουθες:

- `empty`, ελέγχει αν η ουρά είναι άδεια.
- `size`, επιστρέφει το μέγεθος της ουράς.
- `front`, προσπελαύνει το στοιχείο που βρίσκεται στο εμπρός άκρο της ουράς (χωρίς να το αφαιρεί).
- `back`, προσπελαύνει το στοιχείο που βρίσκεται στο πίσω άκρο της ουράς (χωρίς να το αφαιρεί).
- `push`, ωθεί ένα στοιχείο στο πίσω άκρο της ουράς
- `pop`, αφαιρεί το στοιχείο που βρίσκεται στο εμπρός άκρο της ουράς.

Ένα παράδειγμα χρήσης της `std::queue` παρουσιάζεται στη συνέχεια.

```

1 #include <iostream>
2 #include <queue>
3
4 using namespace std;
5
6 int main() {
7     cout << "std::queue" << endl;
8     queue<int> aqueue;
9     cout << "1. Enqueue 10 items" << endl;
10    for (int i = 51; i < 60; i++) {
11        cout << i << " ";
12        aqueue.push(i);
13    }
14    cout << endl << "2. Dequeue 5 items" << endl;
15    for (int i = 0; i < 5; i++) {
16        cout << aqueue.front() << " ";
17        aqueue.pop();
18    }
19    cout << endl << "3. Enqueue 3 items" << endl;
20    for (int i = 61; i <= 63; i++) {
21        cout << i << " ";
22        aqueue.push(i);
23    }

```

```

24 while (!aqueue.empty()) {
25     cout << aqueue.front() << " ";
26     aqueue.pop();
27 }
28 cout << endl;
29 }

```

Κώδικας 4: Παράδειγμα χρήσης της std::queue (stl\_queue\_example.cpp)

```

1 std::stack example
2 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
3 Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

```

## 5 Παραδείγματα

### 5.1 Παράδειγμα 1

Να γραφεί πρόγραμμα που να δέχεται μια φράση ως παράμετρο γραμμής εντολών και να εμφανίζει το εάν είναι παλινδρομική ή όχι. Μια φράση είναι παλινδρομική όταν διαβάζεται η ίδια από αριστερά προς τα δεξιά και από δεξιά προς τα αριστερά.

```

1 #include <iostream>
2 #include <stack>
3
4 using namespace std;
5 // examples of palindromic sentences:
6 // SOFOS, A MAN A PLAN A CANAL PANAMA, AMORE ROMA, LIVE NOT ON EVIL
7 int main(int argc, char **argv) {
8     if (argc != 2) {
9         cerr << "Usage examples: " << endl;
10        cerr << "\t\t" << argv[0] << " SOFOS" << endl;
11        cerr << "\t\t" << argv[0] << " \"A MAN A PLAN A CANAL PANAMA\"" << endl;
12        exit(-1);
13    }
14    string str = argv[1];
15    stack<char> astack;
16    string str1;
17    for (char c : str)
18        if (c != ' ') {
19            str1 += c;
20            astack.push(c);
21        }
22    string str2;
23    while (!astack.empty()) {
24        str2 += astack.top();
25        astack.pop();
26    }
27    if (str1 == str2)
28        cout << "The sentence " << str << " is palindromic." << endl;
29    else
30        cout << "The string " << str << " is not palindromic." << endl;
31 }

```

Κώδικας 5: Έλεγχος παλινδρομικής φράσης (lab05\_ex1.cpp)

```

1 $ lab05_ex1
2 Usage examples:
3     lab05_ex1 SOFOS
4     lab05_ex1 "A MAN A PLAN A CANAL PANAMA"
5

```

```
6 $ lab05_ex1 "A MAN A PLAN A A CANAL PANAMA"
7 The string A MAN A PLAN A A CANAL PANAMA is not palindromic.
8
9 $ lab05_ex1 "A MAN A PLAN A A CANAL PANAM"
10 The string A MAN A PLAN A A CANAL PANAM is not palindromic.
```

## 5.2 Παράδειγμα 2

Να γραφεί πρόγραμμα που να δέχεται ένα δυαδικό αριθμό ως λεκτικό και να εμφανίζει την ισοδύναμη δεκαδική του μορφή.

```
1 #include <iostream>
2 #include <stack>
3 #include <string> // stoi
4
5 using namespace std;
6 int main() {
7     string bs;
8     stack<int> astack;
9     cout << "Enter a binary number: ";
10    cin >> bs;
11    for (char c : bs) {
12        if (c != '0' && c != '1') {
13            cerr << "use only digits 0 and 1" << endl;
14            exit(-1);
15        }
16        astack.push(c - '0');
17    }
18
19    int sum = 0, x = 1;
20    while (!astack.empty()) {
21        sum += astack.top() * x;
22        astack.pop();
23        x *= 2;
24    }
25    cout << "Decimal: " << sum << endl;
26
27    cout << "Decimal: " << stoi(bs, nullptr, 2); // one line solution :)
28 }
```

Κώδικας 6: Μετατροπή δυαδικού σε δεκαδικό (lab05\_ex2.cpp)

```
1 Enter a binary number: 1010101010101011111100111
2 Decimal: 178958311
3 Decimal: 178958311
```

## 6 Ασκήσεις

1. Να υλοποιηθεί η δομή της ουράς χρησιμοποιώντας αντικείμενα στοίβας (std::stack) και τις λειτουργίες που επιτρέπονται σε αυτά. Υλοποιήστε τις λειτουργίες της ουράς empty, size, enqueue, dequeue και front.
2. Να υλοποιηθεί η δομή της στοίβας χρησιμοποιώντας αντικείμενα ουράς (std::queue) και τις λειτουργίες που επιτρέπονται σε αυτά. Υλοποιήστε τις λειτουργίες της στοίβας empty, size, push, pop και top.

## **Αναφορές**

- [1] Tech Crash Course, C Program to Implement a Stack using Singly Linked List, <http://www.techcrashcourse.com/2016/06/c-program-implement-stack-using-linked-list.html>