Основни структури от данни

Моника Велчева

Какво ще научим

- Какво е структура от данни и защо ни трябва
- Как работят основните структури
- Кога да използваме коя
- Как се анализира ефективност (Big-O)

Какво е структура от данни

Определение

Структура от данни е начин да организираме и съхраняваме данни, така че операциите върху тях да са ефективни.

- Пример: Масив, Стек, Опашка, Списък, Множество, Речник
- Цел: бърз достъп, добавяне и търсене

Масив (Array)

Основна идея

Последователни елементи в паметта, достъпни по индекс.

Достъп до елемент: O(1) **Обхождане:** O(n)

Пример: Масив

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

int main() {
    int a[4] = {5, 8, 12, 20};
    cout << a[2] << "\n"; // 12
}
```

Резултат

12

Свързан списък (List)

Основна идея

Всеки елемент сочи към следващия — елементите не са непременно подредени в паметта.

Достъп: O(n) **Добавяне/изтриване:** O(1)

Пример: Списък

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

int main() {
    list<int> L = {10, 25, 31};
    L.push_front(5);
    L.push_back(47);
    for (int x : L) cout << x << " ";
}
```

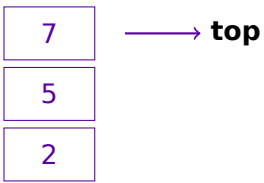
Резултат

```
5 10 25 31 47
```

Стек (Stack)

Основна идея

Последен влязъл — първи излиза (LIFO).



Операции: push, pop, top **Сложност:** O(1)

Пример: Стек

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

int main() {
    stack<int> S;
    S.push(2);
    S.push(5);
    S.push(7);
    cout << S.top() << "\n"; // 7
}
```

Результат

7

Опашка (Queue)

Основна идея

Първи влязъл — първи излиза (FIFO).

Операции: push, pop, front **Сложност:** O(1)

Пример: Опашка

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

int main() {
    queue<int> Q;
    Q.push(2);
    Q.push(5);
    Q.push(7);
    cout << Q.front() << "\n"; // 2
}
```

Резултат

2

Пример: Множество

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

int main() {
    unordered_set<int> S;
    S.insert(5);
    S.insert(10);
    S.insert(5);
    cout << S.count(10) << "\n"; // 1
}
```

Результат

1

Пример: Речник

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

int main() {
    unordered_map<string, int> score;
    score["Alice"] = 95;
    score["Bob"] = 82;
    cout << score["Alice"] << "\n"; // 95
}
```

Резултат

95

Как се пресмята сложност (Big-O)

Идея

Броим колко стъпки извършва алгоритъмът спрямо размера на входа n.

- O(1) — константно време (достъп по индекс)
- O(n) — линейно време (обхождане)
- O(n log n) — сортиране
- O(n^2) — двойни цикли

Обобщение

- Масив — бърз достъп по индекс
- Списък — лесно добавяне и изтриване
- Стек — последен влиза, първи излиза
- Опашка — първи влиза, първи излиза
- Множество — уникални елементи
- Речник — двойки ключ–стойност

Домашна работа

Задача 1: Работа с масив

Напишете програма, която:

- чете n цели числа в масив;
- намира и отпечатва най-голямото и най-малкото.

Домашна работа

Задача 2: Списък от имена

Създайте списък с имена на ученици. Добавете име в началото, премахнете едно от средата и отпечатайте резултата.

Домашна работа

Задача 3: Собствен стек

Имплементирайте свой стек с помощта на масив. Добавете операции: push(x), pop(), top().

Домашна работа

Задача 4: Опашка от клиенти

Напишете програма, която симулира опашка в магазин:

- всеки клиент има име;
- нов клиент се добавя в края;
- когато клиент бъде обслужен, се премахва от началото.

Домашна работа

Задача 5: Уникални числа

Напишете програма, която чете n числа и отпечатва колко от тях са уникални (без повторения).

Домашна работа

Задача 6: Честота на думи

Създайте речник, който брои колко пъти се среща всяка дума в изречение, въведено от потребителя.