Опашка

Библиотека: queue

Синтатсис: queue <type> name;

Опашката е шаблонна[[1]](#footnote-1) структура данни, действаща на принципа FIFO[[2]](#footnote-2). Характерното за тази структура е, че имаме достъп само до стойността в началото й, но добавяме нови елементи в края.

Пример:

queue<string>q;

Често използвани методи и оператори:

|  |  |
| --- | --- |
| .front() | Връща стойността на елемента, намиращ се в началото на опашката. |
| .push(x) | Добавя нов елемент със стойност x в края на опашката. |
| .pop() | Премахва елемента в началото на опашката. |
| .empty() | Проверява дали опашката е празна. Връща true, ако е така, и false в противен случай. |
| = | Оператор за присвояване. |

Приоритетна опашка

Библиотека: queue

Синтатсис: priority\_queue <type> name;

Приоритетната опашка е структура, силно подобна на стандартната опашка с тази разлика, че поддържа наредба на елементите. По подразбиране тази наредба е в низходящ ред, т.е. най-големият се намира в началото (max queue), но може да е и възходящ според дефиницията (min queue).

Пример:

priority\_queue<string>max\_pq; //максимална приоритетна опашка с елементи от тип string

priority\_queue<int, std::vector<int>, std::greater<int> >min\_pq; //минимална приоритетна опашка с елементи от тип int

Често използвани методи и оператори:

|  |  |
| --- | --- |
| .top() | Връща стойността на елемента, намиращ се в началото на опашката. |
| .push(x) | Добавя нов елемент със стойност x в края на опашката. |
| .pop() | Премахва елемента в началото на опашката. |
| .empty() | Проверява дали опашката е празна. Връща true, ако е така, и false в противен случай. |
| = | Оператор за присвояване. |

Дек

Библиотека: deque

Синтатсис: deque <type> name;

Дек[[3]](#footnote-3) (двойна опашка) наричаме шаблонна структура от данни, част от стандартната библиотека STL, която се отличава с достъп до първия и последния елемент в редицата, т.е. може да се добавят и премахват елементи и от двата края. Също като стека и опашката, декът не е поддържа достъп до останалите си елементи.

Пример:

deque<double> d;

Често използвани методи и оператори:

|  |  |
| --- | --- |
| .begin() | Връща итератор към началото на дека. |
| .end() | Връща итератор към края (след последния елемент) на дека. |
| .empty() | Проверява дали декът е празен. Връща true, ако е така, и false в противен случай. |
| .push\_front(x) | Добавя нов елемент със стойност x в началото на дека. |
| .push\_back(x) | Добавя нов елемент със стойност x в края на стека. |
| .pop\_front() | Премахва елемент от началото на дека. |
| .pop\_back() | Премахва елемент от края на дека. |

1. От произволен тип (int, double, vector, …) [↑](#footnote-ref-1)
2. First in – first out. Т.е. първият добавен елемент се премахва първи подред. [↑](#footnote-ref-2)
3. Deque – double ended queue [↑](#footnote-ref-3)