Раздел 1. Введение в C++

Тема 1.1. Простые программы

Рассмотрим несколько примеров программ на C++:

```cpp
#include <iostream> // подключаем ввод-вывод

using namespace std; // заклинание

// точка входа в программу
int main() {
    cout << "Hello world!"; // выводим строку
    return 0; // выход из программы
}
```

```cpp
#include <iostream>

using namespace std;

int main() {
    int n = 0; // объявляем переменную "n"
    cin >> n; // считываем значение в переменную "n"
    int sum = 0;
    for (int i = 0; i < n; i++) { // цикл i = 0 ... (n - 1)
        int x = 0;
        cin >> x;
        sum = sum + x;
    }
    cout << sum;
}
```

```cpp
#include <iostream>

using namespace std;

int my_sum(int x, int y) { // создаем функцию "my_sum"
    int z = x + y;
    return z;
}

int main() {
    int a = 0;
    cin >> a;
    int b = 0;
    cin >> b;
    int c = my_sum(a, b); // вызываем функцию "my_sum"
    cout << c;
    return 0;
}
```

```cpp
#include <iostream>
#include <vector> // подключаем массивы

using namespace std;

int main() {
    int n = 0;
    cin >> n;
    vector<int> arr; // создаем пустой массив "arr"
    for (int i = 0; i < n; i++) {
        int x = 0;
        cin >> x;
        arr.push_back(x); // добавляем число "x" в конец массива
    }
    return 0;
}
```

```cpp
#include <iostream>
#include <vector> // подключаем массивы

using namespace std;

int main() {
    int n = 0;
    cin >> n;
    vector<int> arr(n); // создаем массив "arr" размера "n" (из нулей)
    for (int i = 0; i < n; i++) {
        cin >> arr[i]; // считываем сразу в i-ю ячейку массива
    }
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " "; // выведем элементы массива через пробел
    }
    return 0;
}
```

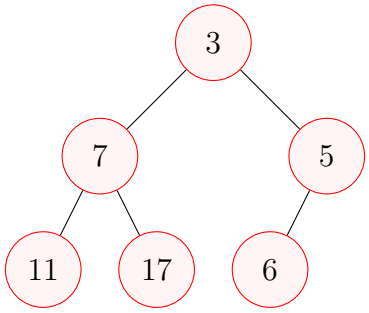
Раздел 2. Сортировки. Бинпоиск. Куча

Тема 2.1. Сортировка слиянием

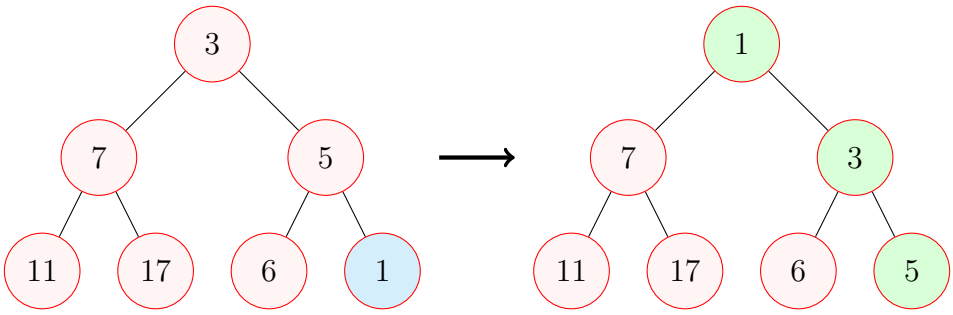
```
vector<int> merge(vector<int> x, vector<int> y) {
    vector<int> ans; int i = 0; int j = 0;
    while (i < x.size() && j < y.size()) {
        if (x[i] < y[j]) {
            ans.push_back(x[i++]);
        } else {
            ans.push_back(y[j++]);
        }
    }
    while (i < x.size()) ans.push_back(x[i++]);
    while (j < y.size()) ans.push_back(y[j++]);
    return ans;
}

vector<int> merge_sort(vector<int> arr) {
    if (arr.size() <= 1) {
        return arr;
    }
    int n = arr.size(); int m = arr.size() / 2;
    vector<int> left; vector<int> right;
    for (int i = 0; i < m; i++) left.push_back(arr[i]);
    for (int i = m; i < n; i++) right.push_back(arr[i]);
    return merge(merge_sort(left), merge_sort(right));
}
```

Тема 2.2. Двоичная куча

- Свойства двоичной кучи:
  - Подвешенное двоичное дерево
  - На i-м слое 2^i вершин (кроме последнего слоя)
  - Последний слой заполнен слева-направо «без пробелов»
  - Верно одно из двух:
    - на всех ребрах написано отношение '$\leqslant$' (*куча по минимуму*)
    - на всех ребрах написано отношение '$\geqslant$' (*куча по максимуму*)



- Операции с двоичной кучей (*по минимуму*):
  - `siftUp(`x`)` — просеять элемент x вверх по дереву
  - `siftDown(`x`)` — просеять элемент x вниз по дереву
  - `add(`x`)` — добавить элемент x в кучу (добавляем x последней вершиной на последнем слое, а затем вызываем `siftUp(`x`)`)



  - `extractMin()` — извлечь текущий минимальный элемент из кучи (свапаем корень с последней вершиной на последнем слое, удаляем последнюю вершину на последнем слое, а затем вызываем `siftDown(root)`)