

Laporan Uas

Algoritma Dan Pemrograman II



Name: Arif Indra Pratama

NIM :231011403713

Class : 03TPLP029

Nama Dosen : Fajar Agung Nugroho

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PAMULANG

1. `#include <iostream>`

`#include <queue>`

```

#include <unordered_map>

using namespace std;

// Struktur node untuk Huffman Tree
struct Node {
    char ch;
    int freq;
    Node *left, *right;
};

// Comparator untuk priority queue
struct Compare {
    bool operator()(Node* l, Node* r) {
        return l->freq > r->freq;
    }
};

// Fungsi untuk membangun Huffman Tree
Node* buildHuffmanTree(const string &text) {
    unordered_map<char, int> freq;
    for (char ch : text) freq[ch]++;

    priority_queue<Node*, vector<Node*>, Compare> pq;
    for (auto pair : freq) {
        Node* node = new Node{pair.first, pair.second, nullptr, nullptr};
        pq.push(node);
    }

    while (pq.size() > 1) {
        Node* left = pq.top(); pq.pop();
        Node* right = pq.top(); pq.pop();
    }
}

```

```

Node* parent = new Node{'\0', left->freq + right->freq, left, right};

pq.push(parent);
}

```

```

return pq.top();
}

```

The screenshot shows a C++ program in Visual Studio. The code is in a file named 'egular Expression.cpp'. It includes headers for `<iostream>`, `<queue>`, and `<unordered_map>`, and uses the `std` namespace. The program defines a `Node` struct for the Huffman tree, a `Compare` struct for the priority queue, and a `buildHuffmanTree` function. The main function calls `buildHuffmanTree` with the input string '10'. The console output shows the input string '10', the encoded string '10', and the decoded string '10'. The process exits after 4.281 seconds with a return value of 0.

```

1 1
0 0
Encoded string: 10
Decoded string: 10

-----
Process exited after 4.281 seconds with return va
lue 0
Press any key to continue . . .

```

Compilation results...

```

Errors: 0
Warnings: 0
Output Filename: C:\Users\LENOVO\Downloads\Regular Expression.exe
Output Size: 2,07019996643066 MiB
Compilation Time: 1,55s

```

2.

The screenshot shows a C++ IDE with a file named `no_1_C.cpp`. The code implements a function `findPairs` that uses a hash set to find two numbers in an array that sum to a target. The `main` function tests this with arrays `{1, 2, 4, 5, 7}` and `{5, 6, 3, 4, 8}` and a target of 9. The output window shows the program's execution, displaying the pairs (4, 5) and (5, 4) and confirming the process exited successfully.

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

void findPairs(int arr1[], int n1, int arr2[], int n2, int target) {
    unordered_set<int> set1(arr1, arr1 + n1);

    for (int i = 0; i < n2; i++) {
        int complement = target - arr2[i];
        if (set1.find(complement) != set1.end()) {
            cout << "(" << complement << ", " << arr2[i] << ")\n";
        }
    }
}

int main() {
    int arr1[] = {1, 2, 4, 5, 7};
    int arr2[] = {5, 6, 3, 4, 8};
    int target = 9;

    cout << "Pasangan bilangan dengan jumlah " << target << ":\n";
    findPairs(arr1, 5, arr2, 5, target);

    return 0;
}
```

Compilation results...
Errors: 0
Warnings: 0
Output Filename: C:\Users\LENOVO\Downloads\no_1_C.exe
Output Size: 1,92695331573486 MiB
Compilation Time: 1,36s

Output window content:
Pasangan bilangan dengan jumlah 9:
(4, 5)
(5, 4)

Process exited after 0.1886 seconds with return value 0
Press any key to continue . . .

3.

The screenshot shows a C++ IDE with a file named `.C.cpp`. The code implements a recursive quicksort function `quickSortFunctional` using vectors to partition the array. The `main` function calls this function on the array `{1, 2, 3, 6, 8, 10}`. The output window shows the sorted array and confirms the process exited successfully.

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

vector<int> quickSortFunctional(const vector<int> &arr) {
    if (arr.size() <= 1) return arr;

    int pivot = arr[arr.size() / 2];
    vector<int> less, equal, greater;

    for (int num : arr) {
        if (num < pivot) less.push_back(num);
        else if (num == pivot) equal.push_back(num);
        else greater.push_back(num);
    }

    auto sortedLess = quickSortFunctional(less);
    auto sortedGreater = quickSortFunctional(greater);

    sortedLess.insert(sortedLess.end(), equal.begin(), equal.end());
    sortedLess.insert(sortedLess.end(), sortedGreater.begin(), sortedGreater.end());
    return sortedLess;
}

int main() {
    // ... (array initialization and call to quickSortFunctional)
}
```

Compilation results...
Errors: 0
Warnings: 0
Output Filename: C:\Users\LENOVO\Downloads\no_1_C.exe
Output Size: 1,8985538482666 MiB
Compilation Time: 1.78s

Output window content:
Hasil Quick Sort: 1 1 2 3 6 8 10

Process exited after 0.1706 seconds with return value 0
Press any key to continue . . .

4.

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <cmath>
```

```
using namespace std;
```

```
// Fungsi untuk mendapatkan digit tertentu
```

```
int getDigit(int number, int digitPlace) {
```

```
    return (number / digitPlace) % 10;
```

```
}
```

```
// Counting Sort untuk Radix Sort
```

```
void countingSort(vector<int>& arr, int digitPlace) {
```

```
    int n = arr.size();
```

```
    vector<int> output(n);
```

```
    int count[10] = {0};
```

```
// Hitung frekuensi digit
```

```
for (int num : arr) {
```

```
    int digit = getDigit(num, digitPlace);
```

```
    count[digit]++;
```

```
}
```

```

// Akumulasi jumlah
for (int i = 1; i < 10; i++) {
    count[i] += count[i - 1];
}

// Bangun array terurut berdasarkan digit
for (int i = n - 1; i >= 0; i--) {
    int digit = getDigit(arr[i], digitPlace);
    output[count[digit] - 1] = arr[i];
    count[digit]--;
}

// Salin hasil ke array asli
arr = output;
}

// Radix Sort
void radixSort(vector<int>& arr) {
    int maxVal = *max_element(arr.begin(), arr.end());
    for (int digitPlace = 1; maxVal / digitPlace > 0; digitPlace *= 10) {
        countingSort(arr, digitPlace);
    }
}

// Quick Sort
void quickSort(vector<int>& arr, int low, int high) {
    if (low < high) {
        int pivot = arr[high];
        int i = low - 1;

        for (int j = low; j < high; j++) {

```

```

        if (arr[j] < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    int pi = i + 1;

    quickSort(arr, low, pi - 1);
    quickSort(arr, pi + 1, high);
}
}

```

// Merge Sort

```

void merge(vector<int>& arr, int left, int mid, int right) {
    int n1 = mid - left + 1, n2 = right - mid;
    vector<int> L(arr.begin() + left, arr.begin() + mid + 1);
    vector<int> R(arr.begin() + mid + 1, arr.begin() + right + 1);

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k++] = L[i++];
        } else {
            arr[k++] = R[j++];
        }
    }
    while (i < n1) arr[k++] = L[i++];
    while (j < n2) arr[k++] = R[j++];
}

```

```

void mergeSort(vector<int>& arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

```

```

int main() {
    vector<int> arr = {170, 45, 75, 90, 802, 24, 2, 66};
    vector<int> radixArr = arr, quickArr = arr, mergeArr = arr;

    cout << "Array asli: ";
    for (int num : arr) cout << num << " ";
    cout << endl;

    // Radix Sort
    radixSort(radixArr);
    cout << "Setelah Radix Sort: ";
    for (int num : radixArr) cout << num << " ";
    cout << endl;

    // Quick Sort
    quickSort(quickArr, 0, quickArr.size() - 1);
    cout << "Setelah Quick Sort: ";
    for (int num : quickArr) cout << num << " ";
    cout << endl;

    // Merge Sort
    mergeSort(mergeArr, 0, mergeArr.size() - 1);
}

```



```

cout << "Setelah Merge Sort: ";

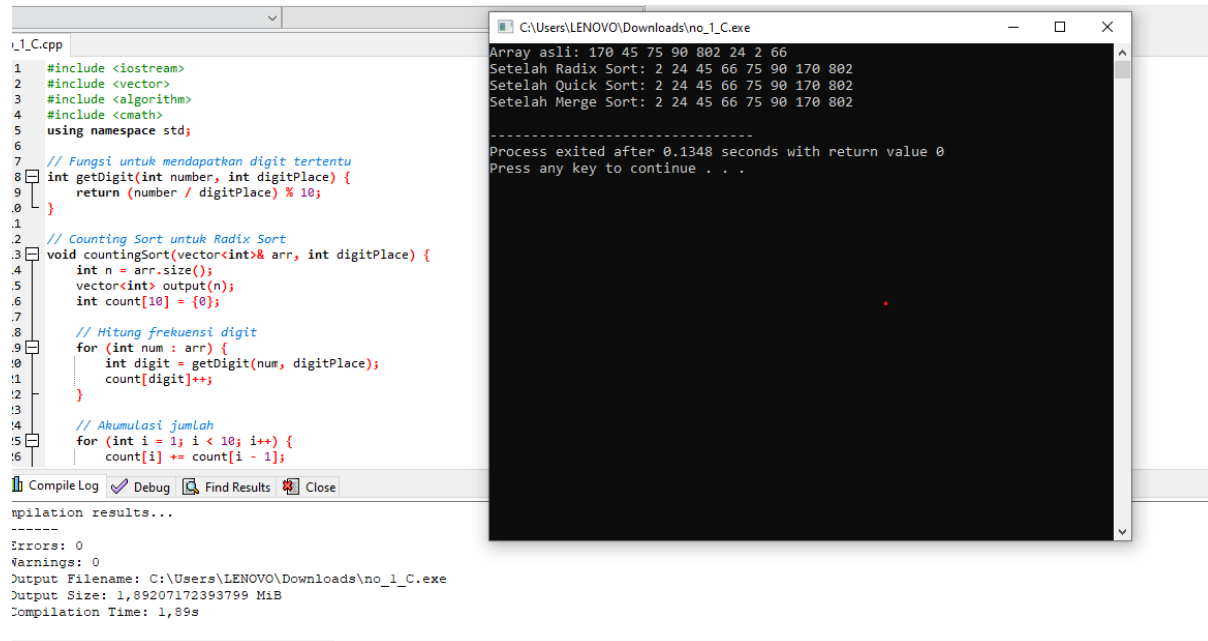
for (int num : mergeArr) cout << num << " ";

cout << endl;

return 0;

}

```



The screenshot shows a C++ IDE with a source file named `no_1_C.cpp` and its execution output in a separate window.

Source Code (no_1_C.cpp):

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <cmath>
5 using namespace std;
6
7 // Fungsi untuk mendapatkan digit tertentu
8 int getDigit(int number, int digitPlace) {
9     return (number / digitPlace) % 10;
10 }
11
12 // Counting Sort untuk Radix Sort
13 void countingSort(vector<int>& arr, int digitPlace) {
14     int n = arr.size();
15     vector<int> output(n);
16     int count[10] = {0};
17
18     // Hitung frekuensi digit
19     for (int num : arr) {
20         int digit = getDigit(num, digitPlace);
21         count[digit]++;
22     }
23
24     // Akumulasi jumlah
25     for (int i = 1; i < 10; i++) {
26         count[i] += count[i - 1];
27     }
28 }

```

Execution Output (no_1_C.exe):

```

Array asli: 170 45 75 90 802 24 2 66
Setelah Radix Sort: 2 24 45 66 75 90 170 802
Setelah Quick Sort: 2 24 45 66 75 90 170 802
Setelah Merge Sort: 2 24 45 66 75 90 170 802
-----
Process exited after 0.1348 seconds with return value 0
Press any key to continue . . .

```

5. #include <iostream>

#include <cmath>

#include <vector>

using namespace std;

// Fungsi rekursif untuk menggambar segitiga

void drawSierpinski(vector<vector<char>>& canvas, int x, int y, int size) {

if (size == 1) { // Base case: ukuran terkecil

canvas[y][x] = '*'; // Gambar titik

return;

}

// Bagian tengah

```

int half = size / 2;

// Rekursi untuk 3 bagian segitiga
drawSierpinski(canvas, x, y, half);    // Segitiga atas
drawSierpinski(canvas, x - half, y + half, half); // Segitiga kiri bawah
drawSierpinski(canvas, x + half, y + half, half); // Segitiga kanan bawah
}

int main() {
    int size = 32; // Ukuran sisi segitiga (harus pangkat 2)
    vector<vector<char>> canvas(size, vector<char>(size * 2, ' ')); // Kanvas

    // Panggil fungsi untuk menggambar Sierpinski Triangle
    drawSierpinski(canvas, size - 1, 0, size);

    // Tampilkan hasil di konsol
    for (const auto& row : canvas) {
        for (char ch : row) cout << ch;
        cout << endl;
    }

}

#include <iostream>
#include <cmath>
#include <vector>
using namespace std;

// Fungsi rekursif untuk menggambar segitiga
void drawSierpinski(vector<vector<char>>& canvas, int x, int y, int size) {
    if (size == 1) { // Base case: ukuran terkecil
        canvas[y][x] = '*'; // Gambar titik
        return;
    }

```

```

}

// Bagian tengah
int half = size / 2;

// Rekursi untuk 3 bagian segitiga
drawSierpinski(canvas, x, y, half);      // Segitiga atas
drawSierpinski(canvas, x - half, y + half, half); // Segitiga kiri bawah
drawSierpinski(canvas, x + half, y + half, half); // Segitiga kanan bawah
}

int main() {
    int size = 32; // Ukuran sisi segitiga (harus pangkat 2)
    vector<vector<char>> canvas(size, vector<char>(size * 2, ' ')); // Kanvas

    // Panggil fungsi untuk menggambar Sierpinski Triangle
    drawSierpinski(canvas, size - 1, 0, size);

    // Tampilkan hasil di konsol
    for (const auto& row : canvas) {
        for (char ch : row) cout << ch;
        cout << endl;
    }
}

```

```

#include <iostream>
#include <cmath>
#include <vector>
using namespace std;

// Fungsi rekursif untuk menggambar segitiga
void drawSierpinski(vector<vector<char>>& canvas, int x, int y, int size) {
    if (size == 1) { // Base case: ukuran terkecil
        canvas[y][x] = '*'; // Gambar titik
        return;
    }

    // Bagian tengah
    int half = size / 2;

    // Rekursi untuk 3 bagian segitiga
    drawSierpinski(canvas, x, y, half); // Segitiga atas
    drawSierpinski(canvas, x - half, y + half, half); // Segitiga kiri
    drawSierpinski(canvas, x + half, y + half, half); // Segitiga kanan
}

int main() {
    int size = 32; // Ukuran sisi segitiga (harus pangkat 2)
    vector<vector<char>> canvas(size, vector<char>(size * 2, ' '));

    // Panggil fungsi untuk menggambar Sierpinski Triangle
}

```

Compile Log Debug Find Results Close

Compilation results...

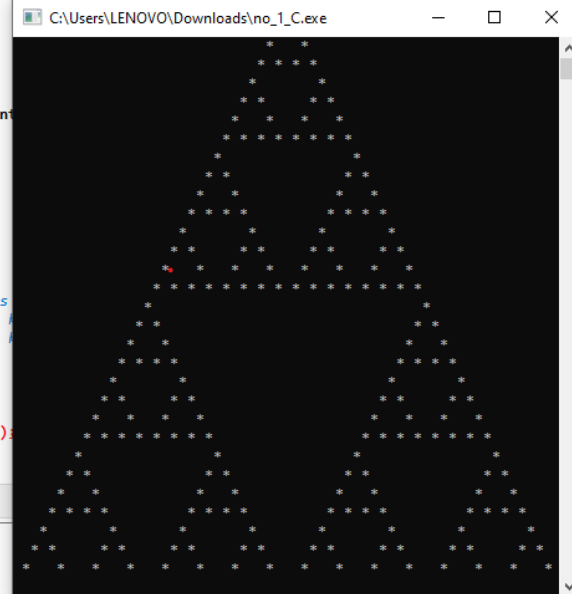
Crosses: 0

Warnings: 0

Output Filename: C:\Users\LENOVO\Downloads\no_1_C.exe

Output Size: 1,87459087371826 MiB

Compilation Time: 6,24s



0 Lines: 35 Length: 1063 Insert Done parsing in 0.015 seconds