Практическая работа №14

Алгоритмы сжатия и кодирования данных.

Автор: Николаев-Аксенов И. С.

Группа: ИКБО-20-19

Код программы:

```
#include <iostream>
#include <vector>
#include <list>
#include <string>
#include <algorithm>
#include <clocale>
#include <map>
#include <iomanip>
#include <queue>
using namespace std;
map<char, string> codes;
map<char, int> freq;

struct MinHeapNode
{
    char data;
    int freq;
    MinHeapNode* left, * right;

    MinHeapNode(char data, int freq)
    {
        left = right = NULL;
        this->data = data;
        this->freq = freq;
    }
};


struct compare
{
    bool operator()(MinHeapNode* l, MinHeapNode* r)
    {
        return (l->freq > r->freq);
    }

};


void storeCodes(struct MinHeapNode* root, string str)
{
    if (root == NULL)
        return;
    if (root->data != '$')
        codes[root->data] = str;
    storeCodes(root->left, str + "0");
    storeCodes(root->right, str + "1");
}


priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap;
template<typename T> void print_queue(T& q,int size)
{
    priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> q1=q;
    priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> q2 = q;

    cout << "Алфавит: ";
```

```
58.     while(!q.empty())
59.     {
60.         if (q.top() != NULL)
61.             cout << setw(4) << q.top()->data << " ";
62.
63.         q.pop();
64.     }
65.
66.     cout << endl << "Количество вхождений: ";
67.     while (!q1.empty())
68.     {
69.         if (q1.top() != NULL)
70.             cout << setw(4) << q1.top()->freq << " ";
71.
72.         q1.pop();
73.     }
74.
75.     cout << endl << "Вероятность: ";
76.     while (!q2.empty())
77.     {
78.         if (q2.top() != NULL)
79.             cout << setw(4) << float(q2.top()->freq)/size << " ";
80.
81.         q2.pop();
82.     }
83.
84.     cout << '\n';
85. }
86. void HuffmanCodes(int size)
87. {
88.     struct MinHeapNode* left, * right, * top;
89.
90.     for (map<char, int>::iterator v = freq.begin(); v != freq.end(); v++)
91.         minHeap.push(new MinHeapNode(v->first, v->second));
92.     priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap2=minHeap;
93.     print_queue(minHeap2,size);
94.
95.     while (minHeap.size() != 1)
96.     {
97.         left = minHeap.top();
98.         minHeap.pop();
99.         right = minHeap.top();
100.          minHeap.pop();
101.          top = new MinHeapNode('$', left->freq + right->freq);
102.          top->left = left;
103.          top->right = right;
104.          minHeap.push(top);
105.      }
106.      storeCodes(minHeap.top(), "");
107. }
108.
109. void calcFreq(string str, int n)
110. {
111.     for (int i = 0; i < str.size(); i++)
112.         freq[str[i]]++;
113. }
114.
115. string haffman_code(string input)
116. {
117.     string encodedString;
118.     for (auto i : input)
119.         encodedString += codes[i];
120.     return encodedString;
121. }
122.
123. void print_table(string str)
124. {
125.     cout << "Алфавит: ";
126.     for (auto item : freq)
```

```
127.         cout << setw(4) << item.first << " ";
128.
129.     cout << endl << "Количество вхождений: ";
130.     for (auto item : freq)
131.         cout << setw(4) << item.second << " ";
132.
133.     cout << endl << "Вероятность: ";
134.     for (auto item : freq) {
135.         cout.setf(std::ios::fixed);
136.         cout << setprecision(2) << float(item.second)/str.length() << " ";
137.     }
138.
139.     cout << endl;
140.     cout << endl;
141. }
142.
143. void sviaz_codov()
144. {
145.     for (auto s : codes)
146.         cout << s.first << ": " << s.second << endl;
147. }
148.
149. int dec2bin(int num)
150. {
151.     int bin = 0, k = 1;
152.     while (num)
153.     {
154.         bin += (num % 2) * k;
155.         k *= 10;
156.         num /= 2;
157.     }
158.     return bin;
159. }
160.
161. string ascii_code(string input) {
162.     string asci = "";
163.     for (int i = 0; i < input.size(); ++i) {
164.         asci += to_string(dec2bin((int(input[i]))));
165.     }
166.     return asci;
167. }
168.
169. void results(string a) {
170.     cout << "Коды символов: " << endl; sviaz_codov();
171.     cout << "Код по Хаффману: " << haffman_code(a) << endl;
172.     cout << "Длина кода по алгоритму Хаффмана: " << haffman_code(a).length() << endl;
173.     cout << "Код по ASCII: " << ascii_code(a) << endl;
174.     cout << "Длина кода по ASCII: " << ascii_code(a).length() << endl;
175.     cout << "Дисперсия " << ((float)haffman_code(a).length() / ascii_code(a).length())
    << endl;
176. }
177.
178. int main()
179. {
180.     setlocale(LC_ALL, "Russian");
181.     string str = "nikolaevaxenov ivan sergeevich";
182.     calcFreq(str, str.length());
183.     print_table(str);
184.     HuffmanCodes(str.length());
185.     results(str);
186.     return 0;
187. }
```

Результат выполнения программы:

```
Алфавит:          a    c    e    g    h    i    k    l    n    o    r    s    v    x
Количество вхождений:    2    3    1    5    1    1    3    1    1    3    2    1    1    4    1
Вероятность: 0.07 0.10 0.03 0.17 0.03 0.03 0.10 0.03 0.03 0.10 0.07 0.03 0.03 0.13 0.03

Алфавит:    c    h    x    r    s    g    k    l         o    i    n    a    v    e
Количество вхождений:    1    1    1    1    1    1    1    1    2    2    3    3    3    4    5
Вероятность: 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.07 0.07 0.10 0.10 0.10 0.13 0.17
Коды символов:
 : 1001
a: 000
c: 10000
e: 111
g: 10101
h: 10001
i: 010
k: 11000
l: 11001
n: 001
o: 1101
r: 10111
s: 10100
v: 011
x: 10110
Код по Хаффману: 00101011000110111001000111011000101101110011101011100101001100000110011010011110111101011111
10110101000010001
Длина кода по алгоритму Хаффмана: 110
Код по ASCII: 11011101101001110101111011111101100110000111001011110110110000111110001100101110111011011111110
10100000110100111101101100001110111010000011100111100101111001011100101111011011010011100011110100
00
Длина кода по ASCII: 208
Дисперсия 0.53
```