| | |
|---|---|
| **課題：I111 3rd Report** | **言語：C++(Console Application)** |
| **氏名：GAO, Yuwei** | |
| **学生番号：s1910092** | **提出日：2019/05/06** |

① 100 までの素数 25 個からなる昇順配列 data がある．
レポート課題２のアルゴリズム 3 を用いている．
プログラムの 4-27 行

② ここから，2 分探索法で 15, 31, 91 を探すプログラムを書け．P10 を参考にしてよい．
28-45 行

③ この data を hash(x) = x % 30 というハッシュ関数を用い長さ 30 の配列に格納せよ．
P18 を参考にしてよい．
54-94 行

④ add 関数の htb[j] != 0 のチェック回数をカウントし，その 25 回分の合計を求めよ．
70 行と 86-94 行

⑤ 余裕があれば, m=30 ではなく, 28,29,50 の場合についても調べ, その結果を考察せよ．
96-124 行

実行結果は最後のところにある

```cpp
#include <iostream>
#include <math.h>
#include <vector>
void getPrimeNumber(short n, std::vector<short>& primenumbers)//レポート課題2のアルゴリズム3を用いている
{
    std::vector<bool> prime;
    for (short i = 0; i <= n; i++)
    {
        prime.push_back(true);
    }
    for (short i = 2, k = sqrt(n); i <= k; i++)
    {
        for (short j = i, l;; j++)
        {
            l = i * j;
            if (l > n)break;
            prime[l] = false;
        }
    }
    for (short i = 2; i <= n; i++)
    {
        if (prime[i])
        {
            primenumbers.push_back(i);
        }
    }
}
int find(short x, std::vector<short>& primenumbers)//2分探索法
{
    std::cout << "Find " << x << "\n";
    short left = 0;
    short right = primenumbers.size() - 1;
    do {
        int mid = (left + right) / 2;
        std::cout << "[" << left << "," << right << "] mid=" << mid << "\n";
        if (x == primenumbers[mid]) {
            std::cout << "Found!" << "\n\n";
            return mid;
        }
        if (x < primenumbers[mid]) right = mid - 1;
        else left = mid + 1;
    } while (left <= right);
    std::cout << "[" << left << "," << right << "] left>right" << "\n" << "Not exist." << "\n\n";
    return -1;
}
int main()
{
    std::vector<short> primenumbers;
    getPrimeNumber(100, primenumbers);
    find(15, primenumbers);
    find(31, primenumbers);
    find(91, primenumbers);

    std::vector<short> htb;
```

```
55      static short m;
56      static int count;
57      static class HashAlgorithm//ハッシュ法のアルゴリズム
58      {
59      public:
60          static void init(std::vector<short>& htb)
61          {
62              for (short i = 1; i <= m; i++)
63              {
64                  htb.push_back(0);
65              }
66          }
67          static short hash(short x) { return x % m; }
68          static void add(short x, std::vector<short>& htb) {
69              short j = hash(x);
70              while (count++ && htb[j] != 0)//htb[j] != 0 のチェック回数をカウント
71              {
72                  j = (j + 1) % m;
73              }
74              htb[j] = x;
75          }
76          static int find(short x, std::vector<short>& htb) {
77              short j = hash(x);
78              while (htb[j] != 0) {
79                  if (htb[j] == x) return j;
80                  j = (j + 1) % m;
81              }
82              return -1;
83          }
84      };
85      //m = 30、25回分の合計
86      m = 30;
87      count = 0;
88      htb.clear();
89      HashAlgorithm::init(htb);
90      for (auto& i : primenumbers)
91      {
92          HashAlgorithm::add(i, htb);
93      }
94      std::cout << "m=" << m << " count=" << count << "\n";
95      //m = 28、25回分の合計
96      m = 28;
97      count = 0;
98      htb.clear();
99      HashAlgorithm::init(htb);
100     for (auto& i : primenumbers)
101     {
102         HashAlgorithm::add(i, htb);
103     }
104     std::cout << "m=" << m << " count=" << count << "\n";
105     //m = 29、25回分の合計
106     m = 29;
107     count = 0;
108     htb.clear();
109     HashAlgorithm::init(htb);
```

```
110     for (auto& i : primenumbers)
111     {
112         HashAlgorithm::add(i, htb);
113     }
114     std::cout << "m=" << m << " count=" << count << "\n";
115     //m = 50、25回分の合計
116     m = 50;
117     count = 0;
118     htb.clear();
119     HashAlgorithm::init(htb);
120     for (auto& i : primenumbers)
121     {
122         HashAlgorithm::add(i, htb);
123     }
124     std::cout << "m=" << m << " count=" << count << "\n";
125     return 0;
126 }
127 /**実行結果////////////////////
128 Find 15
129 [0, 24] mid = 12
130 [0, 11] mid = 5
131 [6, 11] mid = 8
132 [6, 7] mid = 6
133 [6, 5] left > right
134 Not exist.
135
136 Find 31
137 [0, 24] mid = 12
138 [0, 11] mid = 5
139 [6, 11] mid = 8
140 [9, 11] mid = 10
141 Found!
142
143 Find 91
144 [0, 24] mid = 12
145 [13, 24] mid = 18
146 [19, 24] mid = 21
147 [22, 24] mid = 23
148 [24, 24] mid = 24
149 [24, 23] left > right
150 Not exist.
151
152 m = 30 count = 56
153 m = 28 count = 51
154 m = 29 count = 51
155 m = 50 count = 31
156 **////////////////////////////
```