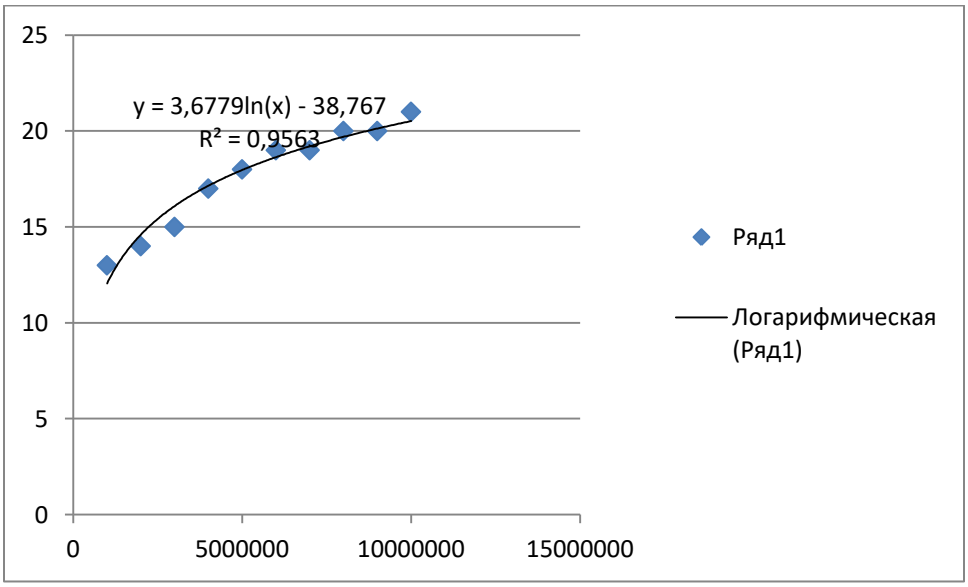
```
// Организация таймирования операции поиска в дереве
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#include <time.h>

// Tree node – зависит от типа дерева
struct Node{
      int key;
      Node *left,           // left subtree
            *right,         // right subtree
            *parent;        // parent node
};

// функции ввода:
int getInt(int *);    // целого

// функции для работы с деревом
int insert(Node **, int);  // вставка элемента
Node *find(Node *, int);   // поиск элемента по ключу
void delTree(Node **);     // освобождение памяти
Node *newNode(int, Node *); // создание нового элемента дерева

// Альтернативы меню для организации диалога
const char *msgs[] = { "0. Quit", "1. Add", "2. Find", "3. Delete", "4. Show",
"5. Timing" };  // список альтернатив
const int NMsgs = sizeof(msgs) / sizeof(msgs[0]);     // количество альтернатив

// функции для организации диалога;
// при обнаружении конца файла возвращают 0
int dialog(const char *msgs[], int);  // выбор номера альтернативы
int D_Add(Node **),   // вставка элемента в таблицу
D_Find(Node **),// поиск элемента в таблице
D_Delete(Node **),    // удаление элемента из таблицы
D_Show(Node **),// вывод содержимого таблицы
D_Timing(Node **);    // таймирование

// массив указателей на функции - для реализации выбора функции
// порядок перечисления функций должен соответствовать
// порядку указания альтернатив в списке альтернатив
int(*fptr[])(Node **) = { NULL, D_Add, D_Find, D_Delete, D_Show, D_Timing };

int main()
{
      Node *root = ...;     //инициализация зависит от типа дерева
      int rc;
      while (rc = dialog(msgs, NMsgs))
      if (!fptr[rc](&root))
            break;
      printf("That's all. Bye!\n");
      delTree(&root);
      return 0;
}
```

```
// таймирование. Аргумент функции не нужен, так как создается собственное дерево
int D_Timing(Node **)
{
      Node *root = &EList;
      int n = 10, key[10000], k, cnt = 1000000, i, m;
      clock_t first, last;
      srand(time(NULL));
      while (n-- > 0){
            for (i = 0; i < 10000; ++i)
                  key[i] = rand() * rand();
            for (i = 0; i < cnt; ){
                  k = rand() * rand();
                  if (insert(&root, k))
                        ++i;
            }
            m = 0;
            first = clock();
            for (i = 0; i < 10000; ++i)
            if (find(root, key[i]))
                  ++m;
            last = clock();
            printf("%d items was found\n", m);
            printf("test #%d, number of nodes = %d, time = %d\n", 10 - n, (10 -
n)*cnt, last - first);
      }
      delTree(&root);
      return 1;
}

// далее – все необходимые функции для работы с деревом
```

```
/*
Результаты тестирования
0. Quit
1. Add
2. Find
3. Delete
4. Show
5. Timing
Make your choice: -->

5
95 items was found
test #1, number of nodes = 1000000, time = 13
214 items was found
test #2, number of nodes = 2000000, time = 14
301 items was found
test #3, number of nodes = 3000000, time = 16
402 items was found
test #4, number of nodes = 4000000, time = 17
478 items was found
test #5, number of nodes = 5000000, time = 18
604 items was found
test #6, number of nodes = 6000000, time = 19
671 items was found
test #7, number of nodes = 7000000, time = 19
788 items was found
test #8, number of nodes = 8000000, time = 20
907 items was found
test #9, number of nodes = 9000000, time = 20
922 items was found
test #10, number of nodes = 10000000, time = 21

That's all. Bye!
Для продолжения нажмите любую клавишу . . .
*/
```

Иллюстрации из Excel: зависимость времени поиска от количества узлов дерева



Иллюстрации из Excel: зависимость времени поиска от высоты дерева