Программирование на языке C

Урок 11. Рекурсия, быстрая сортировка, двоичный поиск

Рекурсия

Рекурсия – это прием программирования, при котором функция или программа вызывает сама себя непосредственно или косвенно.

Например, вычисление факториала легко можно представить рекурсивной функцией

$$!N = N * !(N-1)$$

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

long int Fact(long int N)
{
	// если произведена попытка вычислить факториал нуля
	if (N < 1) return 0;
		// если вычисляется факториал единицы
		// именно здесь производится выход из рекурсии
		else if (N == 1) return 1;
	// любое другое число вызывает функцию заново с формулой N-1
	else return N * Fact(N-1);
}

void main()
{
	long number = 5;
	// первый вызов рекурсивной функции
	long result = Fact(number);
	cout << "Result " << number << "! is - " << result << "\n";
}
```

Быстрая сортировка

1. Из массива выбирается некоторый опорный элемент a[i].
2. Запускается функция разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], слева от него, а все ключи, большие, либо равные a[i] — справа, теперь массив состоит из двух частей, причем элементы левой меньше элементов правой.
3. Если в подмассиве более двух элементов, рекурсивно запускаем для них ту же функцию.
4. В конце получится полностью отсортированная последовательность.

```
void quickSortR(T a[], long N) {
    // На входе - массив a[], a[N] - его последний элемент.
    // поставить указатели на исходные места
    long i = 0, j = N;
    T temp, p;
    p = a[ N/2 ]; // центральный элемент

    // процедура разделения
    do {
        while ( a[i] < p ) i++;
        while ( a[j] > p ) j--;
        if (i <= j) {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
            i++;
            j--;
        }
    } while ( i <= j );
    // рекурсивные вызовы, если есть, что сортировать
    if ( j > 0 ) quickSortR(a, j);
    if ( N > i ) quickSortR(a+i, N-i);
}
```

Двоичный поиск

1. Ищем срединный элемент и сравниваем с искомым значением
2. Если значения равны, поиск завершаем
3. Если искомое больше срединного элемента, то повторяете 1 шаг с правой половиной массива
4. Если искомое меньше срединного элемента, то повторяете 1 шаг с левой половиной

! Работает только для упорядоченных массивов.

Алгоритм описан для массива отсортированного по возрастанию

```
int BinarySearch (int A[], int Lb, int Ub, int Key)
{
    int M;
    while(1) {
        M = (Lb + Ub)/2;
        if (Key < A[M])
            Ub = M - 1;
        else if (Key > A[M])
            Lb = M + 1;
        else
            return M;
        if (Lb > Ub)
            return -1;
    }
}
```