Лекция 3: Задача поиска и функции (процедуры) в языке Си

Д. А. Караваев

Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича

Факультет РТС, Кафедра РОС

Факультатив «Программирование в ЦОС»

Осень 2019

28.10.2019 Санкт-Петербург

Функции (процедуры)

Функция (процедура) - выделенный блок кода, у которого есть набор входных и выходных аргументов, а так же (возвращающее) значение.

Замечание: Строго говоря в (*процедурном*) программировании функция, не является математической функцией в полном смысле этого слова, так как процессу исполнения (вычисления) функции могут сопутствовать побочные эффекты (например, ввод/вывод). В *функциональном* программировании это не совсем так (*Haskell, Scala, Erlang, Lisp*).

Примеры функций

Fuction Sum(a : массив, N : длина a):

```
    sum ← 0
    for n ← 0 to N − 1 do
        sum ← sum + a[n]
    end
    return sum
```

Fuction Search(a : массив, N : длина a, x : искомая величина):

```
    for n ← 0 to N − 1 do
        if a[n] == x then
            return n
        end
    end
    return −1
```

Псевдокод: Функции вычисления суммы элементов массива и поиска значения в массиве.

Вычислительная сложность

Временная сложность: функция оценивающая число арифметических и/или логических операций, выполняемых программой для получения результата, в зависимости от объема входных данных.
Асимптотическая сложность: оценка порядка роста сложности алгоритма (N - объем входных данных):

1. $O(g(N))$ - рост сложности ограничен функцией $g(N)$ сверху;
2. $\Omega(g(N))$ - рост сложности ограничен функцией $g(N)$ снизу;
3. $\Theta(g(N))$ - рост сложности ограничен функцией $g(N)$ сверху и снизу;

Примеры: Сложность алгоритма поиска и суммы: $O(N)$ и $\Theta(N)$ соответственно.
Подробнее - Томас Кормен «*Алгоритмы. Вводный курс*».

Рекурсия

Определение: описание работы алгоритма через самого себя, но на меньшем объёме данных.

Сумма: Запишем рекусривное определение суммы от N элементов через сумму от $N-1$ элементов (сумма 0 элементов есть 0):

$$\sum_{n=0}^{N-1} a[n] = a[N-1] + \sum_{n=0}^{N-2} a[n] \tag{1}$$

$$\sum_{n=0}^{0} a[n] \equiv 0$$

Замечание: Рекурсивное определение функции имеет схожесть с доказательством методом математической индукции (в обратном порядке): есть начальное условие (нулевое утверждение) и рекурсивное определение (индукционный переход) (https://www.mccme.ru/free-books/shen/shen-induction.pdf).

Алгоритм бинарного поиска

Если в задаче поиска массив отсортирован в порядке возрастания: $\forall n$: $a[n] > x$, *то* x *не может находиться правее от* n, *или* $a[n] < x$ *- не может находиться левее.*

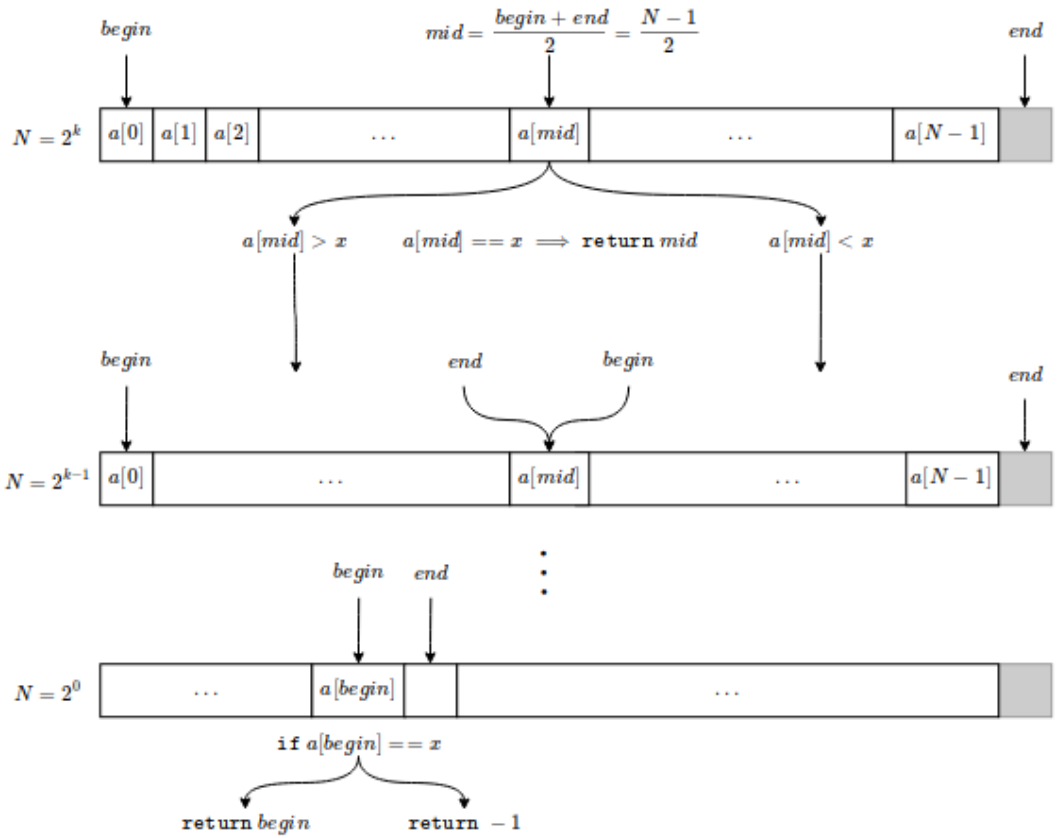

Рис.: Визуализация алгоритма. Сложность $O(k) = O(log_2 N)$

Псевдокод алгоритма бинарного поиска

Fuction BSearch(a : массив, x : искомая величина, $begin$, end):

```
    if begin − end == 1 then
        if a[begin] == x then
            return begin
        else
            return end
        end
    end
    mid ← (begin+end)/2
    if a[mid] == x then
        return mid
    else if a[mid] > x then
        return BSearch (a, x, begin, mid)
    else
        return BSearch (a, x, mid, end)
    end
```

Функции в языке Си

```c
/* Общее определение функции в Си: */
TR function(T1 arg1, T2 arg2, ..., TN argN)
{
    /*!
     * Это называется телом функции, здесь
     * описывается алгоритм в терминах языках Си.
     * Переменные, определенные здесь, не видны в
     * других функциях (это почти всегда верно).
     * TR - (псевдо)тип возвращающего значения;
     * Ti - (псевдо)тип i-ого аргумента;
     * argi - i-ый аргумент функции.
     */
    return /* Вернуть значение некоторого выражение. */;
}
TR ret = function(arg1, arg2, ..., argN); /* Вызов функции. */
```

Примеры функций

```c
/* Сумма двух элементов: */
int32_t sum2(int32_t x, int32_t y)
{
    return x + y;
}
/* Вызов: */
int32_t sum = sum2(5, 14);
/* Отношение двух элементов: */
float div2(float x, float y)
{
    return x / y;
}
/* Вызов: */
float div = div((float)sum, 4);
```

Функция main

```c
/*!
 * В Си есть специальная функция, имя которой зарезервировано.
 * Это функци main, с которой начинается работа программы.
 * Таким образом если Вы хотите, чтобы Ваша функция была
 * исполнена, то её необходимо вызвать прямо или косвенно из
 * функции main.
 */
int main( )
{
    /* Тело main == суть самой программы. */
    /* main должна возвращать 0 - это индикатор
     * успешного завершения программы. */
    return 0;
}
```

Задания

В файле проекта `Search/source/main.c`

1. Определить функцию суммирования целочисленного элементов массива;
2. Определить функцию поиска значения в целочисленном массиве;
3. Определить рекурсивно функцию суммы элементов целочисленного массива;
4. Определить функцию бинарного поиска.

Замечание: Для этого вам придётся создать массив в теле функции `main`, который будет передаваться в качестве аргумента во все эти функции.

Замечание: Для сборки используйте те же команды, что и для проекта `Convolution`. Для проверки результатов воспользуйтесь функцией `printf(...)`, которая печатает значения в терминал: http://www.cplusplus.com/reference/cstdio/printf/.

Спасибо за внимание!