# Лекция 7: Указатели на функцию в языке Си и создание проектов

Д. А. Караваев

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

Факультет РТС, Кафедра РОС

Факультатив «Программирование в ЦОС»

Осень 2019

16.12.2019 Санкт-Петербург

# Функция как параметр

Многие алгоритмы можно сделать зависящими от *функционального* параметра во избежание **дублирования** кода:

- Алгоритм сортировки: функция сравнения (*предикат*) $\Longrightarrow$ сортировка в возрастающем/убывающем порядке;
- Операция `Reduce`: "обобщённое" суммирование (с использование бинарной функции);
- Операция `Map`: преобразование массива по определенном правилу (унарной функции);
- Обобщённый цикл `ForEach`;
- ...

**Вопрос**: Как передать функцию в качестве аргумента другой функции?

# Указатели на функцию

```c
/* Синтаксис: <возвращаемый тип> (* <имя>)(<типы аргументов>): */
/* Некоторая функция: */
float sum(float a, float b)
{
    return a + b;
}
/* Объявление указателя на функцию: */
float (*sum_another)(float, float) = sum;
/* Указатель на функцию sum_over - псевдоним функции sum! */
/* Пример использования: */
float x = sum_another(1.0, 2.0);
printf("(sum == sum_another) == %d", (x == sum(1.0, 2.0)));
/* Результат: (sum == sum_another) == 1. */
```

# Передача указателя на функцию

```c
/* Функция в качестве аргумента: */
void merge_sort(int *a, size_t N, int *(comparator)(int, int));
/* Объявляем "сравниватели": */
int less(int lhs, int rhs)
{
    return a < b;
}
int more(int lhs, int rhs)
{
    return a > b;
}
/* Сортировка по убыванию: */
merge_sort(a, N, less);
/* Сортировка по возрастанию: */
merge_sort(a, N, more);
```

# Функциональный тип

```c
/* Тип функции с заданными аргументами и значением (!интерфейс!): */
typedef float *(binary_op)(float, float);
/* Функции удовлетворяющие типу binary_op: */
float sum(float a, float b)  { /* ... */ }
float prod(float a, float b) { /* ... */ }
/* "Свернуть" массив по правилу: */
float reduce(float *a, size_t N, binary_op bop, float base)
{
    float ret = base;
    for (size_t i = 0; i < N; ++i)
    {
        ret = bop(ret, a[i]);
    }
    return ret;
}
```

# Вычисление определённого интеграла

Необходимо вычислить определённый интеграл на промежутке $[a, b]$ от произвольной функции $f(x)$:

$$S = \int_a^b f(x)\mathrm{d}x \tag{1}$$

Для этого пользуются геометрической интерпретацией интеграла как площади кривой под функцией $f(x)$.

Существует большое множество различный методов *численного* интегрирования (Самарский А. А. Введение в численные методы).

# Метод прямоугольников

Интервал $[a, b]$ разбивается на *равномерную сетку* с шагом $d$:

$$d = (b - a)/N,\ \{x_i = a + id\},\ i \in \{0, \ldots, N - 1\},$$

$$S \approx \int_a^b f(x)\mathrm{d}x = d \sum_{i=0}^{N-1} f(x_i).$$

Т.е. функция $f(x)$ *интерполируется* на интервале $(x_i, x_{i+1})$ константами (горизонтальными прямыми) (0-ой порядок аппроксимации).

# Метод трапеций

Интервал $[a, b]$ разбивается на *равномерную сетку* с шагом $d$:

$$d = (b - a)/N,\ \{x_i = a + id\},\ i \in \{0, \ldots, N - 1\},$$

$$S \approx \int_a^b f(x)\mathrm{d}x = d \sum_{i=0}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2}.$$

Функция $f(x)$ *интерполируется* на интервале $(x_i, x_{i+1})$ прямыми (1-ый порядок аппроксимации).

# Метод Симпсона (парабол)

Интервал $[a, b]$ разбивается на *равномерную сетку* с шагом $d$:

$$d = (b-a)/N,\ \{x_i = a + id\},\ i \in \{0, \ldots, N-1\},$$

$$S \approx \int_a^b f(x)\mathrm{d}x = \frac{d}{6}\sum_{i=0}^{N-1}\left(f(x_i) + 4f\left(\frac{f(x_i)+f(x_{i+1})}{2}\right) + f(x_{i+1})\right).$$

Функция $f(x)$ *интерполируется* на интервале $(x_i, x_{i+1})$ параболами (2-ой порядок аппроксимации).

# Метод Монте-Карло

Пусть $u$ - случайная велечина, распределённая равномерно на $[a, b] \implies f(u)$ так же случайная величина со средним:

$$\mathrm{E}\{f(u)\} = \int_a^b f(x)\varphi(x)\mathrm{d}x = \frac{1}{b-a}\int_a^b f(x)\mathrm{d}x,$$

$$\int_a^b f(x)\mathrm{d}x = (b-a)\mathrm{E}\{f(u)\} \approx \frac{b-a}{N}\sum_{i=0}^{N-1} f(\hat{u}_i),$$

где $\hat{u}_i,\ i \in \{0, \ldots, N-1\}$ - конкретные значения случайной величины $u$ (выборка размера $N$).

# Задание

Реализовать один из 4-х методов численного интегрирования в виде функции на языке Си, которая принимает $f(x)$ в качестве параметра:

```
/* Тип функции: */
typedef float *(unary_function)(float, float);
/* Прототип функции: */
float num_integral(float a, float b, unary_function f, size_t N);
```

**NB**: Для того чтобы реализовать вычисление методом Монте-Карло необходимо воспользоваться стандартной фукнцией `rand`, подключив заголовочный файл `stdlib`:
http://www.cplusplus.com/reference/cstdlib/rand/.

# Создание проекта

Создадим проект для решения задачи численного интегрирования:

```
# Создадим папку проекта:
mkdir Integral && cd Integration
# Создадим папки для файлов с кодом:
mkdir include source
# Создадим файлы:
touch source/main.c source/integral.c include/integral.h CMakelists.txt
```

В языке Си файл с иходным кодом деляться на два типа:

1. Заголовочные (`*.h`) - для объявления типов и прототипов (интерфейсный файлы);
2. Исходные (`*.c`) - для реализации функций (файлы реализации).

# Пример: заголовочный/исходный файл

Файл include/integral.h:

```c
#include <stdlib.h> /* Подключение стадартного заголовочного файла. */
/* Тип функции: */
typedef float *(unary_function)(float, float);
/* Прототип функции: */
float num_integral(float a, float b, unary_function f, size_t N);
```

Файл source/integral.c:

```c
#include "integral.h" /* Подключение заголовочного файла. */
/* Определение функции: */
float num_integral(float a, float b, unary_function f, size_t N)
{
    /* ... */
}
```

# Пример: файл с функцией main

Теперь функцией num_integral можно воспользоваться в функции main:

```c
#include <stdio.h> /* Подключение стадартного заголовочного файла. */
#include <math.h> /* Заголовочный файл с математическим функциями. */

#include "integral.h" /* Подключение заголовочного файла. */

int main()
{
    float S = num_integral(0, 3.14 * 5, sin, 1000);
    printf("S{sin(x)} = %f\n", S);
    return 0;
}
```

По подобному принципу формируются *библиотеки* с исходным кодом, у которых есть некоторое предназначение (библиотека для работы с изображениями OpenCV).

# Пример: файл сборки CMakeLists.txt

Исходный код в исполняемый файл преобразуют компилятор + линковщик. Прямая работа с этими программами неудобна при большом числе файлов в проекте, поэтому существует системы сборки, такие как cmake: https://cmake.org/.

---

```
# Создание проекта для сборки с именем:
PROJECT(Integral C)
# Переменная с именем исполняемого файла:
SET(EXEC_TARGET integral)
# Добавить папку с заголовочными файлами:
INCLUDE_DIRECTORIES(${CMAKE_CURRENT_SOURCE_DIR}/include)
# Переменные со списком заголовочных и исходных файлов:
SET(HEADERS ./include/integral.h)
SET(SOURCES ./source/integral.c ./source/main.c)
# Команда сборки исполняемого файла:
ADD_EXECUTABLE(${EXEC_TARGET} ${SOURCES})
```

---

# Спасибо за внимание!