

Работа на ЭВМ и программирование (группа 112)

Занятие 5

Контактная информация

- Шундеев Александр Сергеевич
- alex.shundeev@gmail.com
- <http://group112.github.io/sem1.html>

Электронная почта

- Тема письма

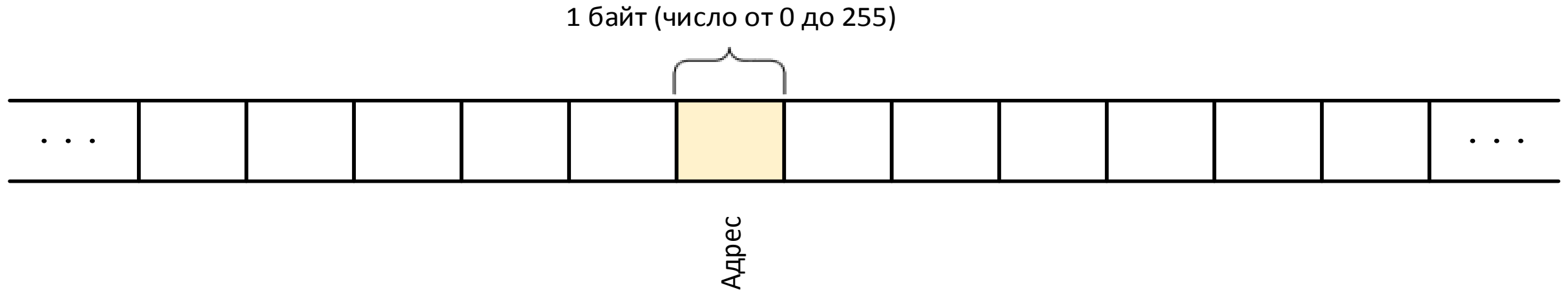
- 112 Фамилия Имя Отчество
- 112 Фамилия Имя

- Пример

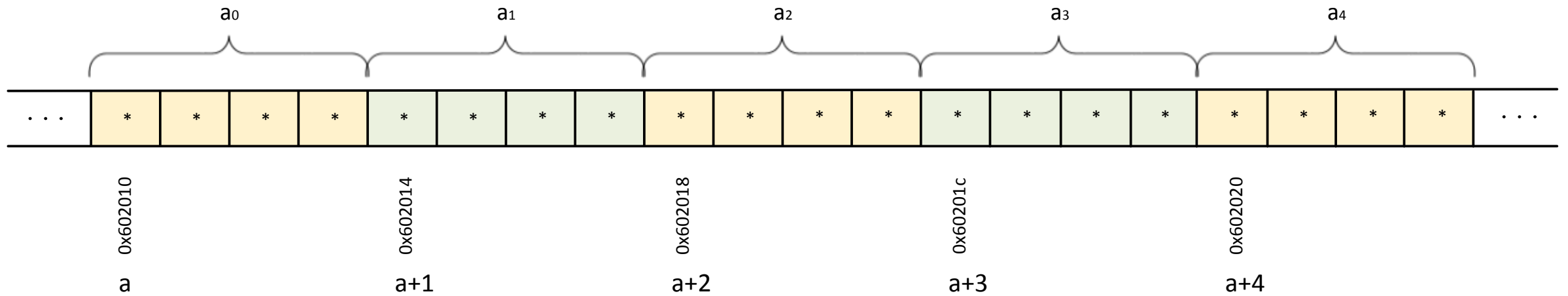
- 112 Иванов Иван Иванович
- 112 Иванов Иван

Арифметика указателей

Виртуальное адресное пространство



Размещение массива в памяти



Арифметика указателей

Переменная типа указатель на T (например, `int` или `double`)

```
T *p;
```

```
p = <адрес>;
```

Переменная типа `int`

```
int n;
```

```
n = <число>;
```

Арифметика указателей

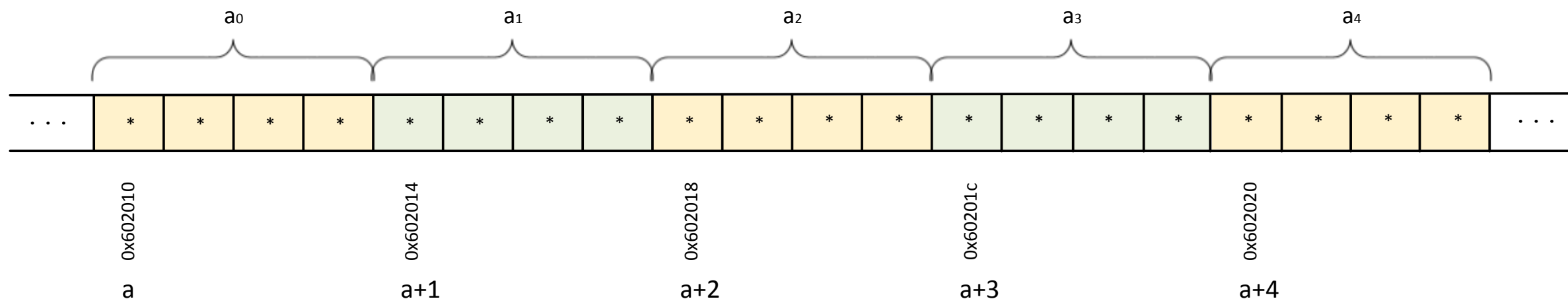
Значением выражения (сложение указателя и числа)

$$p + n$$

является адрес

$$\text{<адрес>} + \text{<число>} * \text{sizeof}(T)$$

Размещение массива в памяти



`int *a = 0x602010;` (в настоящих программах так делать не надо)

`// a + 1 == 0x602014`

`(0x602010 + 1 * sizeof(int))`

`// a + 2 == 0x602018`

`(0x602010 + 2 * sizeof(int))`

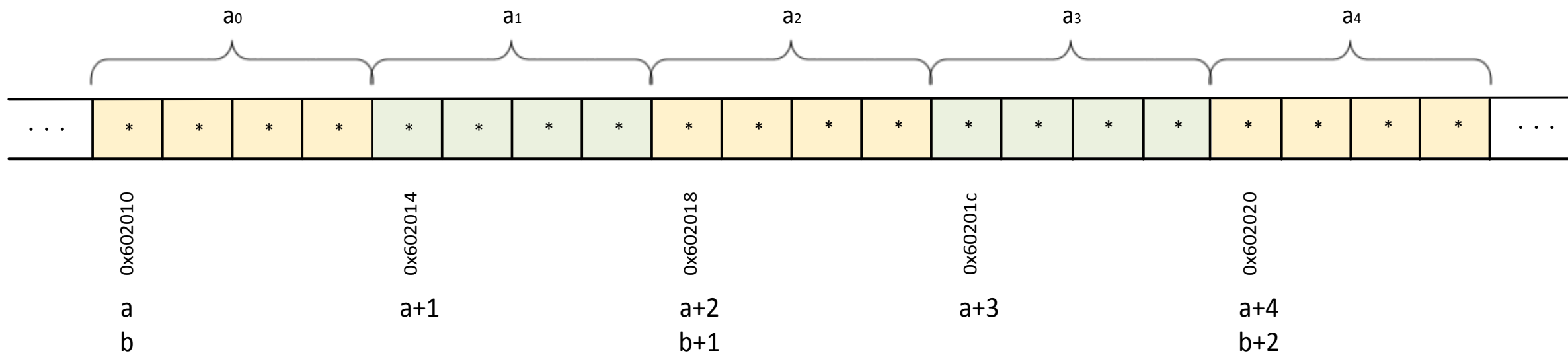
`// a + 3 == 0x60201c`

`(0x602010 + 3 * sizeof(int))`

`// a + 4 == 0x602020`

`(0x602010 + 4 * sizeof(int))`

Размещение массива в памяти



`double *b = 0x602010;` (в настоящих программах так делать не надо)

`// b + 1 == 0x602018`

`// b + 2 == 0x602020`

`(0x602010 + 1 * sizeof(double))`

`(0x602010 + 2 * sizeof(double))`

Размещение массива в памяти

a - адрес памяти, выделенной под массив

Адрес i -го элемента массива

$$a + i$$

i -й элемент массива

$$*(a + i) \quad (\text{или } a[i])$$

Массивы

Начало

Входной файл input.txt

Вариант 1 (по умолчанию)

$n \ a_0 \ a_1 \ \dots \ a_{n-1}$

Вариант 2

$a_0 \ a_1 \ \dots \ a_{n-1}$

Входной файл input.txt

Вариант 1 (по умолчанию)

$n \ a_0 \ a_1 \ \dots \ a_{n-1}$

размер
массива

элементы
массива

Вариант 2

$a_0 \ a_1 \ \dots \ a_{n-1}$

элементы
массива

Работа с массивами

Массивы создаются динамически

- Функция `malloc` (выделение памяти)
- Функция `free` (освобождение памяти)

Выделение / освобождение памяти осуществляется внутри функции `main`

Перед выходом из функции `main` (завершением программы) должна быть явно освобождена ранее выделенная память

Работа с массивами

Внимание!

Если не оговорено особо, то память выделяется только под хранение обрабатываемого массива

Функция обработки массива

```
void fun(double *a, int n);
```

- `a` - адрес памяти, выделенной под хранение массива
- `n` - размер массива
- Могут быть и дополнительные параметры


Структура программы (1 файл)

```
#include <stdlib.h>
#include <stdio.h>

void fun(double *a, int n);

int main(void)
{
    ...
    fun(a, n);
    ...
}

void fun(double *a, int n)
{
    ...
}
```



Структура программы (2 файла)

```
// f1.c
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
void fun(double *a, int n);
```

```
int main(void)
```

```
{
```

```
    ...
```

```
    fun(a, n);
```

```
    ...
```

```
}
```

```
// f2.c
```

```
#include <stdio.h>
```

```
void fun(double *a, int n);
```

```
void fun(double *a, int n)
```

```
{
```

```
    ...
```

```
}
```

Команда компиляции

```
gcc -Wall -Wextra -Wfloat-equal -Werror -pedantic -std=c99 f1.c f2.c -o prog
```

Пример программы работы с массивами

Вариант, когда во входном файле указывается длина массива

Структура программы

```
// f1.c
```

```
#include <stdlib.h>
#include <stdio.h>
```

```
double average(double *a, int n);
```

```
int main(void)
{
    ...
    average(a, n);
    ...
}
```

```
// f2.c
```

```
double average(double *a, int n);
```

```
double average(double *a, int n)
{
    ...
}
```

Комментарий. Программа считывает из входного файла числовую последовательность и сохраняет ее в массив. Далее, вычисляется среднее арифметическое значение элементов массива, которое печатается в выходной файл.

Текст программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
```

```
    if(fscanf(fi, "%d", &n) != 1)
    {
        fprintf(stderr, "Can't read parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    if(n < 1)
    {
        fprintf(stderr, "Wrong parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    a = malloc(sizeof(double) * n);
    if(!a)
    {
        fprintf(stderr, "Memory allocation error ...\n");
        fclose(fi);
        return -1;
    }
```

Текст программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```


Текст программы (файл f2.c)

```
// f2.c

double average(double *a, int n);

double average(double *a, int n)
{
    double s = 0.;
    int i;

    for(i = 0; i < n; i++)
        s += a[i];

    return s / n;
}
```

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Переменные `fi` и `fo` - указатели на открытые файлы.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
}
```

Комментарий. Переменные: **a** - указатель на область памяти, выделенную под хранения массива; **r** - вычисляемая характеристика; **x** - вспомогательная (для определения конца файла); **n** - размер массива; **i** - индекс текущего элемента массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
```

```
    if(fscanf(fi, "%d", &n) != 1)
    {
        fprintf(stderr, "Can't read parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    if(n < 1)
    {
        fprintf(stderr, "Wrong parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    a = malloc(sizeof(double) * n);
    if(!a)
    {
        fprintf(stderr, "Memory allocation error ...\n");
        fclose(fi);
        return -1;
    }
```

Комментарий. Открытие входного файла [input.txt](#).

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
```

```
    if(fscanf(fi, "%d", &n) != 1)
    {
        fprintf(stderr, "Can't read parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    if(n < 1)
    {
        fprintf(stderr, "Wrong parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    a = malloc(sizeof(double) * n);
    if(!a)
    {
        fprintf(stderr, "Memory allocation error ...\n");
        fclose(fi);
        return -1;
    }
```

Комментарий. Проверка успешности открытия входного файла [input.txt](#).

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
}
```

Комментарий. Если произошла ошибка открытия файла.

Печатается сообщение об ошибке в `stderr`. Программа завершается и возвращает статус **-1** (ошибка).

Tect

```
$ rm -f input.txt; ./prog; echo $?
```

```
Can't open input.txt ...
```

```
255
```

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Чтение размера массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Если произошла ошибка чтения размера массива. Печатается сообщение об ошибке в [stderr](#). Освобождаются ресурсы (заккрытие входного файла). Программа завершается и возвращает статус **-1** (ошибка).

Tect

```
$ rm -f input.txt; touch input.txt; ./prog; echo $?
```

Can't read parameter `n` ...

255

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
```

```
    if(fscanf(fi, "%d", &n) != 1)
    {
        fprintf(stderr, "Can't read parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    if(n < 1)
    {
        fprintf(stderr, "Wrong parameter `n` ...\n");
        fclose(fi);
        return -1;
    }

    a = malloc(sizeof(double) * n);
    if(!a)
    {
        fprintf(stderr, "Memory allocation error ...\n");
        fclose(fi);
        return -1;
    }
```

Комментарий. Проверка корректности задания размера массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Если размер массива был задан некорректно. Печатается сообщение об ошибке в `stderr`. Освобождаются ресурсы (закрытие входного файла). Программа завершается и возвращает статус `-1` (ошибка).

Tect

```
$ echo "-1" > input.txt; ./prog; echo $?
```

Wrong parameter `n` ...

255

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Выделение памяти под хранение массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Проверка успешности выделения памяти. В случае успеха функция `malloc` возвращает ненулевой адрес.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
if(fscanf(fi, "%d", &n) != 1)
{
    fprintf(stderr, "Can't read parameter `n` ...\n");
    fclose(fi);
    return -1;
}

if(n < 1)
{
    fprintf(stderr, "Wrong parameter `n` ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
}
```

Комментарий. Если произошла ошибка выделения памяти. Печатается сообщение об ошибке в [stderr](#). Освобождаются ресурсы (заккрытие входного файла). Программа завершается и возвращает статус **-1** (ошибка).

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
    r = average(a, n);  
  
    free(a);  
  
    fo = fopen("output.txt", "w");  
    if(!fo)  
    {  
        fprintf(stderr, "Can't open output.txt ...\n");  
        return -1;  
    }  
  
    fprintf(fo, "%f", r);  
  
    fclose(fo);  
  
    return 0;  
}
```

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
    r = average(a, n);  
  
    free(a);  
  
    fo = fopen("output.txt", "w");  
    if(!fo)  
    {  
        fprintf(stderr, "Can't open output.txt ...\n");  
        return -1;  
    }  
  
    fprintf(fo, "%f", r);  
  
    fclose(fo);  
  
    return 0;  
}
```

Комментарий. На каждой итерации цикла осуществляется попытка считать *i*-й элемент массива.

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Если произошла ошибка чтения очередного элемента массива. Печатается сообщение об ошибке в [stderr](#). Освобождаются ресурсы (заккрытие входного файла, освобождение памяти). Программа завершается и возвращает статус **-1** (ошибка).

Tect

```
$ echo "5 1 2 3 4" > input.txt; ./prog; echo $?
```

Can't read element ...

255

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
    r = average(a, n);  
  
    free(a);  
  
    fo = fopen("output.txt", "w");  
    if(!fo)  
    {  
        fprintf(stderr, "Can't open output.txt ...\n");  
        return -1;  
    }  
  
    fprintf(fo, "%f", r);  
  
    fclose(fo);  
  
    return 0;  
}
```

Комментарий. Проверка конца файла (выполнение пробного чтения).

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Если файл содержит в конце «мусор». Печатается сообщение об ошибке в `stderr`. Освобождаются ресурсы (заккрытие входного файла, освобождение памяти). Программа завершается и возвращает статус `-1` (ошибка).

Тест

```
$ echo "5 1 2 3 4 5 6" > input.txt; ./prog; echo $?
```

Wrong input data ...

255

```
$ echo "5 1 2 3 4 5 abc" > input.txt; ./prog; echo $?
```

Wrong input data ...

255

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```


Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
    r = average(a, n);  
  
    free(a);  
  
    fo = fopen("output.txt", "w");  
    if(!fo)  
    {  
        fprintf(stderr, "Can't open output.txt ...\n");  
        return -1;  
    }  
  
    fprintf(fo, "%f", r);  
  
    fclose(fo);  
  
    return 0;  
}
```

Выполнение программы (файл f2.c)

```
// f2.c

double average(double *a, int n);

double average(double *a, int n)
{
    double s = 0.;
    int i;

    for(i = 0; i < n; i++)
        s += a[i];

    return s / n;
}
```

Комментарий. Переменные: **s** - сумма элементов массива; **i** - индекс текущего элемента массива.

Выполнение программы (файл f2.c)

```
// f2.c

double average(double *a, int n);

double average(double *a, int n)
{
    double s = 0.;
    int i;

    for(i = 0; i < n; i ++)
        s += a[i];

    return s / n;
}
```

Комментарий. Цикл по всем элементам массива. На каждой итерации подсчитываемая сумма увеличивается на значение текущего элемента массива.

Выполнение программы (файл f2.c)

```
// f2.c

double average(double *a, int n);

double average(double *a, int n)
{
    double s = 0.;
    int i;

    for(i = 0; i < n; i++)
        s += a[i];

    return s / n;
}
```

Комментарий. Функция возвращает среднее арифметическое значений элементов массива.

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
    r = average(a, n);  
  
    free(a);  
  
    fo = fopen("output.txt", "w");  
    if(!fo)  
    {  
        fprintf(stderr, "Can't open output.txt ...\n");  
        return -1;  
    }  
  
    fprintf(fo, "%f", r);  
  
    fclose(fo);  
  
    return 0;  
}
```

Комментарий. Освобождение памяти, ранее выделенной под хранение массива.

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Открытие выходного файла [output.txt](#).

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Проверка успешности открытия выходного файла [output.txt](#).

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Если произошла ошибка открытия файла.

Печатается сообщение об ошибке в `stderr`. Программа завершается и возвращает статус `-1` (ошибка).

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Программа печатает результат своей работы в выходной файл [output.txt](#).

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
    r = average(a, n);  
  
    free(a);  
  
    fo = fopen("output.txt", "w");  
    if(!fo)  
    {  
        fprintf(stderr, "Can't open output.txt ...\n");  
        return -1;  
    }  
  
    fprintf(fo, "%f", r);  
  
    fclose(fo);  
  
    return 0;  
}
```

Выполнение программы (файл f1.c)

```
for(i = 0; i < n; i++)
    if(fscanf(fi, "%lf", a + i) != 1)
    {
        fprintf(stderr, "Can't read element ...\n");
        fclose(fi);
        free(a);
        return -1;
    }

if(fscanf(fi, "%lf", &x) != EOF)
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    free(a);
    return -1;
}

fclose(fi);
```

```
    r = average(a, n);

    free(a);

    fo = fopen("output.txt", "w");
    if(!fo)
    {
        fprintf(stderr, "Can't open output.txt ...\n");
        return -1;
    }

    fprintf(fo, "%f", r);

    fclose(fo);

    return 0;
}
```

Комментарий. Программа завершается и возвращает статус 0 (успех).

Tect

```
$ echo "5 1 2 3 4 5" > input.txt; ./prog; echo $?; cat output.txt
```

0

3.000000

Пример программы работы с массивами

Вариант, когда во входном файле не указывается длина массива

Текст программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Текст программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%lf", r);

fclose(fo);

return 0;
}
```

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Переменные `fi` и `fo` - указатели на открытые файлы.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Переменные: **a** - указатель на область памяти, выделенную под хранения массива; **r** - вычисляемая характеристика; **x** - вспомогательная (для подсчета размера массива); **n** - размер массива; **i** - индекс текущего элемента массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Открытие входного файла [input.txt](#).

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Проверка успешности открытия входного файла [input.txt](#).

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Если произошла ошибка открытия файла.

Печатается сообщение об ошибке в `stderr`. Программа завершается и возвращает статус `-1` (ошибка).

Tect

```
$ rm -f input.txt; ./prog; echo $?
```

Can't open input.txt ...

255

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Подсчет числа элементов массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Проверка: файл пуст или файл содержит «мусор».

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Если файл пуст или содержит «мусор». Печатается сообщение об ошибке в [stderr](#). Освобождаются ресурсы (заккрытие входного файла). Программа завершается и возвращает статус **-1** (ошибка).

Tect

```
$ rm -f input.txt; touch input.txt; ./prog2; echo $?
```

Wrong input data ...

255

```
$ echo "abc" > input.txt; ./prog2; echo $?
```

Wrong input data ...

255

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Выделение памяти под хранение массива.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Проверка успешности выделения памяти. В случае успеха функция `malloc` возвращает ненулевой адрес.

Выполнение программы (файл f1.c)

```
// f1.c

#include <stdlib.h>
#include <stdio.h>

double average(double *a, int n);

int main(void)
{
    FILE *fi, *fo;
    double *a, r, x;
    int n, i;

    fi = fopen("input.txt", "r");
    if(!fi)
    {
        fprintf(stderr, "Can't open input.txt ...\n");
        return -1;
    }
}
```

```
for(n = 0;; n++)
    if(fscanf(fi, "%lf", &x) != 1)
        break;

if(!n || !feof(fi))
{
    fprintf(stderr, "Wrong input data ...\n");
    fclose(fi);
    return -1;
}

a = malloc(sizeof(double) * n);
if(!a)
{
    fprintf(stderr, "Memory allocation error ...\n");
    fclose(fi);
    return -1;
}
```

Комментарий. Если произошла ошибка выделения памяти. Печатается сообщение об ошибке в [stderr](#). Освобождаются ресурсы (заккрытие входного файла). Программа завершается и возвращает статус **-1** (ошибка).

Выполнение программы (файл f1.c)

```
rewind(fi);
```

```
for(i = 0; i < n; i ++)  
    fscanf(fi, "%lf", a + i);
```

```
fclose(fi);
```

```
r = average(a, n);
```

```
free(a);
```

```
fo = fopen("output.txt", "w");
```

```
if(!fo)  
{  
    fprintf(stderr, "Can't open output.txt ...\n");  
    return -1;  
}
```

```
fprintf(fo, "%f", r);
```

```
fclose(fo);
```

```
return 0;
```

```
}
```

Комментарий. Перемещение текущей позиции чтения в начало файла.

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```


Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Освобождение памяти, ранее выделенной под хранение массива.

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Открытие выходного файла [output.txt](#).

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}

fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Проверка успешности открытия выходного файла [output.txt](#).

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}

fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Если произошла ошибка открытия выходного файла.

Печатается сообщение об ошибке в stderr. Программа завершается и возвращает статус -1 (ошибка).

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Программа печатает результат своей работы в выходной файл [output.txt](#).

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Заккрытие выходного файла.

Выполнение программы (файл f1.c)

```
rewind(fi);

for(i = 0; i < n; i++)
    fscanf(fi, "%lf", a + i);

fclose(fi);

r = average(a, n);

free(a);

fo = fopen("output.txt", "w");
if(!fo)
{
    fprintf(stderr, "Can't open output.txt ...\n");
    return -1;
}
```

```
fprintf(fo, "%f", r);

fclose(fo);

return 0;
}
```

Комментарий. Программа завершается и возвращает статус 0 (успех).

Tect

```
$ echo "5 1 2 3 4 5" > input.txt; ./prog; echo $?; cat output.txt
```

0

3.000000

Отладка программы

Отладочная печать

Функция печати массива

```
void print(double *a, int n)
{
    int i;
    printf("n: %d a:", n);
    for(i = 0; i < n; i++)
        printf(" %.2f", a[i]);
    printf("\n");
}
```

Текст программы (файл f1.c)

```
for(i = 0; i < n; i ++)  
    if(fscanf(fi, "%lf", a + i) != 1)  
    {  
        fprintf(stderr, "Can't read element ...\n");  
        fclose(fi);  
        free(a);  
        return -1;  
    }  
  
if(fscanf(fi, "%lf", &x) != EOF)  
{  
    fprintf(stderr, "Wrong input data ...\n");  
    fclose(fi);  
    free(a);  
    return -1;  
}  
  
fclose(fi);
```

```
print(a, n);  
  
r = average(a, n);  
  
free(a);  
...  
}
```

Тест

```
$ echo "5 1 2 3 4 5" > input.txt; ./prog; echo $?; cat output.txt
```

```
n: 5 a: 1.00 2.00 3.00 4.00 5.00
```

```
0
```

```
3.000000
```

Выполнение программы (файл f2.c)

```
// f2.c

#include <stdio.h>

double average(double *a, int n);

double average(double *a, int n)
{
    double s = 0.;
    int i;

    printf("Function average ...\n");

    for(i = 0; i < n; i++)
        s += a[i];

    return s / n;
}
```