

Работа на ЭВМ и программирование (группа 112)

Занятие 7

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

Корректное чтение из stdin

http://group112.github.io/doc/sem1/2018/2018_sem1_stdin.pdf

Проблема (использование scanf)

```
$ ./prog
```

```
Input x1: 11
```

```
Input x2: 12
```

```
Input x3: 13
```

```
Input x4: 14
```

```
Input x5: 15
```

```
Wrong parameter x5 ...
```

```
$
```

Проблема (использование scanf)

```
$ ./prog
```

```
Input x1: 11
```

```
Input x2: 12
```

```
Input x3: 13
```

```
Input x4: 14
```

```
Input x5: 15
```

```
Wrong parameter x5 ...
```

```
$
```

По новой запускать программу и вводить параметры?

Проблема (использование scanf)

```
$ ./prog
```

```
Input x1: 111
```

```
Input x2: 121
```

```
Input x3: 131
```

```
Input x4: 14|
```

```
Input x5: 151
```

```
Wrong parameter x5 ...
```

```
$
```

Проблема (использование scanf)

```
$ ./prog
```

```
Input x1: 111
```

```
Input x2: 121
```

```
Input x3: 131
```

```
Input x4: 14|
```

```
Input x5: 151
```

```
Wrong parameter x5 ...
```

```
$
```

Почему параметр x5 неправильный?

Решение

```
$ ./prog
```

```
Input x1: 111
```

```
Input x2: 121
```

```
Input x3: 131
```

```
Input x4: 14|
```

```
Wrong parameter x4, Please try again...
```

```
Input x4: |41
```

```
Wrong parameter x4, Please try again...
```

```
Input x4: 141
```

```
Input x5: 151
```

```
$
```

Стандартная функция fgetc

```
int fgetc(FILE *fi);
```

Входной параметр:

- **fi** - указатель на открытый файл;

Результат:

- **код символа**, приведенный к типу int
- **EOF**, в случае конца файла или ошибки чтения

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void);
```

Прочитывает все символы из stdin пока не встретится символ перехода на новую строку (или конец файл)

Результат:

- 0 - были прочитаны только пробельные символы
- -1 - иначе

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: ' ', ' ', ' ', 'a', ' ', '\n'

r: -

c: -

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: ' ', ' ', 'a', ' ', '\n'

Итерация: 1

r: 0

c: ' '

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: ' ', 'a', ' ', '\n'

Итерация: 2

r: 0

c: ' '

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: 'a', ' ', '\n'

Итерация: 3

r: 0

c: ' '

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: ' ', '\n'

Итерация: 4

r: 0

c: 'a'

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: ' ', '\n'

Итерация: 4

r: -1

c: 'a'

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin: '\n'

Итерация: 5

r: -1

c: ' '

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin:

Итерация: 6

r: -1

c: '\n'

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin:

Итерация: 6

r: -1

c: '\n'

«Очистка» стандартного потока ввода (stdin)

```
int skip_chars(void) {  
    int r, c;  
  
    for(r = 0;;) {  
        c = fgetc(stdin);  
  
        if(c == '\n' || c == EOF)  
            break;  
  
        if(c != ' ' && c != '\t')  
            r = -1;  
    }  
  
    return r;  
}
```

stdin:

r: -1

c: '\n'

Корректное чтение целого числа

```
int c, x;
...
for(c = 0; !feof(stdin); c = 1) {
    if(c)
        printf("Please, try again ...\n");
    printf("Input x: ");
    if(scanf("%d", &x) != 1) {
        skip_chars();
        continue;
    }
    if(skip_chars() == -1)
        continue;
    break;
}
printf("%d\n", x);
```

Пример

```
$ ./prog
```

```
Input x4: |41
```


Корректное чтение целого числа

```
int c, x;  
...  
for(c = 0; !feof(stdin); c = 1) {  
    if(c)  
        printf("Please, try again ...\n");  
    printf("Input x: ");  
    if(scanf("%d", &x) != 1) {  
        skip_chars();  
        continue;  
    }  
    if(skip_chars() == -1)  
        continue;  
    break;  
}  
printf("%d\n", x);
```

Пример

```
$ ./prog
```

```
Input x4: |41
```

```
Please try again...
```

```
Input x4: 14|
```

Корректное чтение целого числа

```
int c, x;  
...  
for(c = 0; !feof(stdin); c = 1) {  
    if(c)  
        printf("Please, try again ...\n");  
    printf("Input x: ");  
    if(scanf("%d", &x) != 1) {  
        skip_chars();  
        continue;  
    }  
    if(skip_chars() == -1)  
        continue;  
    break;  
}  
printf("%d\n", x);
```

Пример

```
$ ./prog
```

```
Input x4: |41
```

```
Please try again...
```

```
Input x4: 14|
```

```
Please try again...
```

```
Input x4: 141
```

```
141
```

```
$
```

Корректное чтение целого числа

```
int c, x;  
...  
for(c = 0; !feof(stdin); c = 1) {  
    if(c)  
        printf("Please, try again ...\n");  
    printf("Input x: ");  
    if(scanf("%d", &x) != 1) {  
        skip_chars();  
        continue;  
    }  
    if(skip_chars() == -1)  
        continue;  
    break;  
}  
printf("%d\n", x);
```

Пример

```
$ ./prog
```

```
Input x4: |41
```

```
Please try again...
```

```
Input x4: 14|
```

```
Please try again...
```

```
Input x4: 141
```

```
141
```

```
$
```

Сортировка массива

http://group112.github.io/doc/sem1/2019/2019_sem1_sort.pdf

Постановка задачи

Массив a длины n называется **неубывающим**, если

$$a[0] \leq a[1] \leq a[2] \dots \leq a[n-1]$$

Процедура сортировки «переставляет» элементы массива таким образом, чтобы массив стал неубывающим

Входные параметры и результат

\$./sort

Входные параметры и результат

\$./sort

Input n: 50000

Программа тестирует функцию сортировки на двух массивах
(длины n и длины $2*n$)

Входные параметры и результат

\$./sort

Input n: 50000

Input t (r/i/d): r

Тип массивов, на которых тестируется функция сортировки
(r - случайные, i - возрастающие, d - убывающие)

Входные параметры и результат

\$./sort

Input n: 50000

Input t (r/i/d): r

Input c (y/n): n

Проверять корректность результата сортировки
(y - да, n - нет)

Корректность сортировки

```
int check(int *a, int *t, int n);
```

Входные параметры:

- **a** - массив длины **n**, прошедший сортировку
- **t** - копия массива **a** до его сортировки

Результат: **0** - корректна, **-1** - некорректна

Корректность сортировки

Сортировка корректна, если

- массив **a** не убывает
- каждое значение встречается в массивах **a** и **t** одинаковое число раз

Входные параметры и результат

```
$ ./sort
```

```
Input n: 50000
```

```
Input t (r/i/d ): r
```

```
Input c (y/n): n
```

```
t1: 1.817137000000
```

```
t2: 7.159167000000
```

```
t2/t1: 3.939805859437
```

```
$
```

Результат - время сортировки массива
(t1 - длины n, t2 - длины 2*n)

Время сортировки

Сложность алгоритма $O(n^2)$

$$\frac{t_2}{t_1} \approx 4$$

Сложность алгоритма $O(n \log n)$

$$\frac{t_2}{t_1} \approx 2$$

Замер времени выполнения

Замер времени выполнения функции

```
#include <time.h>
```

```
...
```

```
clock_t  s, e;
```

```
double  t;
```

```
...
```

```
s = clock();
```

```
sort(a, n);
```

```
e = clock();
```

```
t = (double)(e - s) / CLOCKS_PER_SEC;
```

Выделение памяти

(несколько массивов)

Множественный вызов malloc (4-е массива)

```
double *a1, *t1, *a2, *t2;
int n;
...
a1 = malloc(n * sizeof(double));
if(!a1) {
    fprintf(stderr, "Memory allocation error\n");
    return -1;
}
t1 = malloc(n * sizeof(double));
if(!t1) {
    fprintf(stderr, "Memory allocation error\n");
    free(a1);
    return -1;
}
a2 = malloc(2 * n * sizeof(double));
if(!a2) {
    fprintf(stderr, "Memory allocation error\n");
    free(a1);
    free(t1);
    return -1;
}
```

```
t2 = malloc(2 * n * sizeof(double));
if(!t2) {
    fprintf(stderr, "Memory allocation error\n");
    free(a1);
    free(t1);
    free(a2);
    return -1;
}
...
free(a1);
free(t1);
free(a2);
free(t2);
```

Однократный вызов malloc (4-е массива)

```
double *a1, *t1, *a2, *t2;
int n;
...
a1 = malloc(6 * n * sizeof(double));
if(!a1) {
    fprintf(stderr, "Memory allocation error\n");
    return -1;
}
t1 = a1 + n;
a2 = t1 + n;
t2 = a2 + 2*n;
...
free(a1);
```

Случайные данные

(заполнение массива случайными значениями)

Заполнение массива случайными значениями

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

```
...
```

```
int *a, n, i;
```

```
...
```

```
srand(0);
```

```
for(i = 0; i < 2 * n; i ++)
```

```
    a[i] = rand();
```

Структура программы

Структура программы

// f1.c

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

```
void sort(int *a, int n);
int check(int *a, int *t, int n);
```

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

// f2.c

```
void sort(int *a, int n);
int check(int *a, int *t, int n);
```

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

```
int check(int *a, int *t, int n)
{
    ...
}
```

Команда компиляции

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