

Практикум на ЭВМ

План, технические моменты и вспоминаем С

Баев А.Ж.

Казахстанский филиал МГУ

27 сентября 2018

1 часть (С)

- 1 Условный оператор, циклы, статические массивы
- 2 Строки, указатели, динамические массивы, структуры
- 3 Динамические структуры, аргументы командной строки, файлы
- 4 Системные вызовы fork, exec, pipe
- 5 Системные вызовы сети

Каждая тема: 10 баллов в классе + 10 баллов дома

2 часть (С)

- 1 Shell
- 2 Тестирующая система
- 3 Игровой клиент-сервер
- 4 Веб-браузер

Каждая тема: 100 баллов (+ ревью)

3 часть (C++)

- 1 Классы, методы
- 2 Перегрузка операторов, полиморфизм
- 3 Наследование
- 4 Шаблоны
- 5 STL

Каждая тема: 10 баллов в классе + 10 баллов дома

4 часть (C++ QT)

- 1 Калькулятор
- 2 Редактор изображений
- 3 Paint
- 4 Арканойд
- 5 Изометрическая игра

Каждая тема: 20 баллов

Вспомогательные инструменты

- 1 bash
- 2 codestyle
- 3 gdb
- 4 Makefile
- 5 github / gitlab (для семестровых работ)

Самая полезная ссылка

<https://ejudge.ru/study/3sem/unix.shtml>

Нельзя написать свой shell, если вы не умеете пользоваться стандартным shell'ом.

Простой материал:

<https://younglinux.info/bash.php>

https://server.179.ru/wiki/?page=Informatika/Komandy_Linux

Оформление кода

```
#include <stdio.h>

int main(void) {
    int input_a, input_b, sum;
    scanf("%d%d", &input_a, &input_b);
    if ((input_a > 0) && (input_b > 0)) {
        sum = input_a + input_b;
        printf("%d\n", sum);
    } else {
        puts("Bad input");
    }
    return 0;
}
```

- 1 Имена переменных.
- 2 Отступы - пробелы.
- 3 Фигурные скобки.

<https://tproger.ru/translations/stanford-cpp-style-guide/>

Вариант 1. Мягкий чекер cpplint (на python). Ставим из репозитория (можно скачать и просто исходник)

```
sudo apt install python3-pip  
pip install cpplint
```

Вариант 2. Строгий чекер checkpatch (на perl). Скачиваем из репозитория.

<https://github.com/torvalds/linux/blob/master/scripts/checkpatch.pl>

Настройка vim

Файл `.vimrc` в домашней директории:

```
set expandtab
set tabstop=4
set shiftwidth=4
set softtabstop=4
set smarttab
set autoindent
```

Обычный режим

```
gcc 01.c -o 01 -lm -Wall -Werror
```

Для отладки

```
gcc 01.c -o 01 -lm -Wall -Werror -g
```

```
gdb 01
```

<https://server.179.ru/tasks/gdb/>

Makefile с компиляцией и проверкой кода

Создаем текстовый файл Makefile в директории с исходниками:

```
%: %.c
    gcc $@.c -o $@ -Wall -Werror -lm
    cpplint --filter=-legal/copyright $@.c
```

Компиляция и проверка кода в файле 01.c:

```
make 01
```

<https://habr.com/post/155201/>

Задача

Дано положительное вещественное число. Найти первую цифру дробной части числа.

```
#include <stdio.h>
int main(void) {
    double input;
    int output;
    scanf("%lf", &input);
    output = (int)(input * 10) % 10;
    printf("%d\n", output);
    return 0;
}
```

Задача

Даны вещественные координаты двух точек $(x_1; y_1)$ и $(x_2; y_2)$. Необходимо найти площадь пересечения квадратов с центрами в данных точках и стороной 1.


```
#include <stdio.h>
#include <cmath.h>
int main(void) {
    double x1, y1, x2, y2;
    double square = 0;
    scanf("%lf %lf", &x1, &y1);
    scanf("%lf %lf", &x2, &y2);

    double dx = fabs(x2 - x1);
    double dy = fabs(y2 - y1);
    if (dx <= 1 && dy <= 1) {
        square = (1 - dx) * (1 - dy);
    }
    printf("%.2lf\n", square);
    return 0;
}
```

Задача

Дано целое число от 1 до 10^{18} . Вывести цифры числа в обратном порядке.

```
#include <stdio.h>

int main(void) {
    int digit;
    long long input;
    scanf("%lld", &input);

    while (input != 0) {
        digit = input % 10;
        printf("%d", digit);
        input /= 10;
    }
    printf("\n");
    return 0;
}
```

Задача

Дана последовательность положительных целых чисел от -10^{100} до 10^{100} , разделенных знаками (+ или -). Ввод заканчивается символом =.

```
...  
    int ans = 0, current, sign = '+';  
    char ch = getchar();  
    do {  
        current = 0;  
        while ('0' <= ch && ch <= '9') {  
            current = 10 * current + (ch - '0');  
            ch = getchar();  
        }  
        if (sign == '+')  
            ans += current;  
        if (sign == '-')  
            ans -= current;  
        sign = ch;  
    } while (sign != '=');  
    printf("%d\n", ans);  
    ...
```

Задача

Дано целое положительное число от 1 до 10^9 . Разложить его на простые множители (с учетом кратности).

Решение 1

```
...  
    int divisor;  
    for (divisor = 2; divisor <= number; divisor++) {  
        while (number % divisor == 0)  
        {  
            printf("%d□", divisor);  
            number /= divisor;  
        }  
    }  
...
```

Решение 2

```
...
int divisor;
for (divisor = 2; divisor * divisor <= number; divisor++)
    while (number % divisor == 0) {
        printf("%d□", divisor);
        number /= divisor;
    }
if (number > 1) {
    printf("%d□", number);
}
...
```


Задача

Посчитать число инверсий в массиве.

```
...  
    int array[1000];  
    int size, i, inversions = 0;  
    scanf("%d", &size);  
    for (i = 0; i < size; i++) {  
        scanf("%d", &array[i]);  
    }  
    int left, right;  
    for (right = 0; right < size; right++) {  
        for (left = 0; left < right; left++) {  
            if (array[left] > array[right]) {  
                inversions++;  
            }  
        }  
    }  
    ...
```

В матрице размера 2×3 (2 строки 3 столбца) заполняются два её угловых элемента:

```
int a[2][3];  
a[0][0] = 1;  
a[1][2] = 2;
```

В данном случае получится матрица следующего вида:

1	???	???
???	???	2

Таблица умножения 10×10 в виде двумерного массива.

```
...  
int mult[10][10];  
for (i = 0; i < n; i++) {  
    for (j = 0; j < n; j++) {  
        mult[i][j] = (i + 1) * (j + 1);  
    }  
}  
...
```

Инициализировать матрицу можно сразу же при объявлении:

```
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

1	2	3
4	5	6

Матрицы хранятся в (одномерной!) памяти по строкам:

Адрес	0x00	0x04	0x08	0x0C	0x10	0x14
Имя	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
Значение	1	2	3	4	5	6

Указатель на массив совпадает с указателем на первый элемент массива:

```
printf("%p\n", &a);           //0x10000000
printf("%p\n", &a[0]);        //0x10000000
printf("%p\n", &a[1]);        //0x1000000C
printf("%p\n", &a[0][0]);     //0x10000000
printf("%p\n", &a[1][0]);     //0x1000000C
```

$a[i][j]$ элемент по адресу a со смещением $(columns * i + j)$. Нет ошибок:

```
printf("%d_", a[0][3]);       //4
printf("%d_", a[0][4]);       //5
printf("%d_", a[0][5]);       //6
printf("%d_", a[1][-1]);      //3
printf("%d_", a[1][-2]);      //2
printf("%d_", a[1][-3]);      //1
```

Очень важный момент многомерных массивов — отличие первой размерности от остальных. Первая размерность может определяться автоматически, а остальные — нет.

```
int a[][3] = {{1, 2, 3}, {4, 5, 6}}; // можно  
int a[2][] = {{1, 2, 3}, {4, 5, 6}}; // нельзя  
int a[][] = {{1, 2, 3}, {4, 5, 6}}; // нельзя
```


Задача

Дано целое положительно n от 1 до 10. Далее 2 матрицы размера $n \times n$ из целых чисел от -1000 до 1000 . Найти произведение матриц.

```
typedef int Matrix[100][100];
int main() {
    Matrix matrix_a, matrix_b, matrix_c;
    int i, j, k, size;
    /* input */
    for (i = 0; i < size; i++) {
        for (j = 0; j < size; j++) {
            c[i][j] = 0;
            for (k = 0; k < size; k++) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    /* output */
    return 0;
}
```

- Разобраться с gdb
- Прочитать про стиль
- Разобраться с cpp lint
- Разобраться с Makefile
- Практика - 10 задач. Сдаем на msu.ejudge.kz