

Практикум на ЭВМ

План, технические моменты и вспоминаем С

Баев А.Ж.

Казахстанский филиал МГУ

02 сентября 2020

1 План на семестр

2 Вспомогательные инструменты

- Командная строка (bash)
- Редактор кода (vim)
- Стиль кода (codestyle checker)
- Отладчик кода (gdb)
- Компиляция и автосборка (Makefile)

3 Вспоминаем C

- Арифметика
- Условный оператор
- Цикл while
- Цикл do while
- Цикл for
- Массивы
- Матрицы.

1 часть (C)

- 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

2 часть (С)

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

Каждая тема: 100 баллов

3 часть (C++)

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

Каждая тема: 10 баллов

4 часть (C++ QT)

- 1 Калькулятор
- 2 Редактор изображений
- 3 Paint
- 4 Арканойд

Каждая тема: 20 баллов

Вспомогательные инструменты

- 1 Командная строка (bash)
- 2 Редактор кода (vim)
- 3 Компиляция и автосборка (Makefile)
- 4 Стилль кода (codestyle checker)
- 5 Отладчик кода (gdb)
- 6 Хостинг для хранения и разработки кода (github)

Командная строка (bash)

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

Классический материал:

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

https://www.opennet.ru/docs/RUS/bash_scripting_guide/

Игровой материал:

<https://overthewire.org/wargames/bandit/bandit1.html>

Начать игру:

```
ssh bandit0@bandit.labs.overthewire.org -p 2220
```


Навигация:

действие	команда
путь до текущей директории	<code>pwd</code>
содержимое текущей директории	<code>ls</code>
содержимое директории DIR	<code>ls DIR</code>
перейти в директорию DIR	<code>cd DIR</code>
перейти на уровень вверх	<code>cd ..</code>
перейти в предыдущую директорию	<code>cd -</code>
перейти в домашнюю директорию	<code>cd</code>

Файлы и директории:

действие	команда
создать файл FILE	<code>touch FILE</code>
удалить файл FILE	<code>rm FILE</code>
создать директорию DIR	<code>mkdir DIR</code>
удалить пустую директорию DIR	<code>rmdir DIR</code>
удалить директорию DIR	<code>rm DIR -r</code>
скопировать SRC в DST	<code>cp SRC DST</code>
переместить (переименовать) SRC в DST	<code>mv SRC DST</code>
получить статистику файла FILE (строки, слова, символы)	<code>wc FILE</code>
сравнить два файла FILE1 и FILE2	<code>diff FILE1 FILE2</code>
найти файл FILE в директории DIR	<code>find DIR -name FILE</code>

Содержимое:

действие	команда
вывести содержимое файла FILE	cat FILE
вывести начало файла FILE	head FILE
вывести конец файла FILE	tail FILE
убрать столбец 3 из вывода FILE	cut FILE -d ' ' -f 3

Потоки:

действие	команда
записать вывод команды cmd в файл FILE	cmd >FILE
направить на ввод команды cmd данные из файла FILE	cmd <FILE
отсортировать вывод команды cmd	cmd sort
убрать дубликаты подряд cmd	cmd uniq
выбрать строки вывода команды cmd с текстом TEXT	cmd grep TEXT
найти текст foo и заменить на bar в вы-	cmd sed 's/foo/bar/g'

Разное:

действие	команда
свободное место в каталоге DIR (англ. disk free)	df DIR
занятое место в каталоге DIR (англ. disk usage)	du DIR
все процессы (англ. processes)	ps -aux
наиболее активные процессы	top
наиболее активные процессы	htop
завершить процесс под номером ID	kill ID
измерить время работы программы cmd	time cmd
установить пакет PKG	sudo apt install PKG
удалить пакет PKG	sudo apt remove PKG
скачать файл url	wget url

В качестве необычных для новичков программ можно посмотреть программы:

действие	пакет	команда
проиграть аудио файл	sox	play audio.mp3
открыть web страницу	lynx	lynx http://msu.kz
распечатать файл на принтере		lpr file.pdf
посмотреть на паровоз	sl	sl

Очень мощным средством являются конвейеры (обозначены символом `|`, которые позволяют вызывать несколько команд обработки текста друг за другом.

```
cat TEXT | tr ' ' '\n' | sort | uniq | wc -l
```

```
echo "print(2/3)" | python3
```

```
for f in *.py; do echo $f:""; \
  head -n 2 $f | tail -n 1; done | grep "=" -B 2
```

Редактор кода (vim)

Файл `.vimrc` в домашней директории:

```
1 set expandtab
2 set tabstop=4
3 set shiftwidth=4
4 set softtabstop=4
5 set smarttab
6 set autoindent
```

Классический материал:

```
sudo apt install vim
vimtutor
```

Игровой материал:

<https://vim-adventures.com>

Стиль кода (codestyle checker)

```
1  #include <stdio.h>
2
3  int main(void) {
4      int input_a, input_b, sum;
5      scanf("%d %d", &input_a, &input_b);
6
7      if ((input_a > 0) && (input_b > 0)) {
8          sum = input_a + input_b;
9          printf("%d\n", sum);
10     } else {
11         puts("Bad input");
12     }
13     return 0;
14 }
```

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

Стиль кода (codestyle checker)

<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>

Отладчик кода (gdb)

Для отладки

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

```
1 gdb 01
```

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

Компиляция и автосборка (Makefile)

Создаем текстовый файл Makefile в директории с исходниками:

```
1 %: %.c
2     gcc $@.c -o $@ -Wall -Werror -lm
3     cpplint --filter=-legal/copyright $@.c
```

Компиляция и проверка кода в файле 01.c:

```
1 make 01
```

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

Задача

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

Решение

```
1 #include <stdio.h>
2 int main(void) {
3     double input;
4     int output;
5     scanf("%lf", &input);
6     output = (int)(input * 10) % 10;
7     printf("%d\n", output);
8     return 0;
9 }
```

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.

Решение

```
1  #include <stdio.h>
2  #include <cmath.h>
3  int main(void) {
4      double x1, y1, x2, y2;
5      double square = 0;
6      scanf("%lf_%lf", &x1, &y1);
7      scanf("%lf_%lf", &x2, &y2);
8
9      double dx = fabs(x2 - x1);
10     double dy = fabs(y2 - y1);
11     if (dx <= 1 && dy <= 1) {
12         square = (1 - dx) * (1 - dy);
13     }
14     printf("%.2lf\n", square);
15     return 0;
16 }
```

Задача

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

Решение

```
1  #include <stdio.h>
2
3  int main(void) {
4      int digit;
5      long long input;
6      scanf("%lld", &input);
7
8      while (input != 0) {
9          digit = input % 10;
10         printf("%d", digit);
11         input /= 10;
12     }
13     printf("\n");
14     return 0;
15 }
```

Задача

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

Решение

```
1  ...
2      int ans = 0, current, sign = '+';
3      char ch = getchar();
4      do {
5          current = 0;
6          while ('0' <= ch && ch <= '9') {
7              current = 10 * current + (ch - '0');
8              ch = getchar();
9          }
10         if (sign == '+')
11             ans += current;
12         if (sign == '-')
13             ans -= current;
14         sign = ch;
15     } while (sign != '=');
16     printf("%d\n", ans);
17     ...
```

Задача

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

Решение 1

```
1  ...
2  int divisor;
3  for (divisor = 2;
4      divisor <= number;
5      divisor++)
6  {
7      while (number % divisor == 0)
8      {
9          printf("%d□", divisor);
10         number /= divisor;
11     }
12 }
13 ...
```

Решение 2

```
1  ...
2  int divisor;
3  for (divisor = 2;
4      divisor * divisor <= number;
5      divisor++)
6  {
7      while (number % divisor == 0) {
8          printf("%d□", divisor);
9          number /= divisor;
10     }
11     if (number > 1) {
12         printf("%d□", number);
13     }
14     ...
```

Задача

Посчитать число инверсий в массиве.

Решение

```
1  ...
2      int array[1000];
3      int size, i, inversions = 0;
4      scanf("%d", &size);
5      for (i = 0; i < size; i++) {
6          scanf("%d", &array[i]);
7      }
8      int left, right;
9      for (right = 0; right < size; right++) {
10         for (left = 0; left < right; left++) {
11             if (array[left] > array[right]) {
12                 inversions++;
13             }
14         }
15     }
16     ...
```


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

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

В данном случае получится матрица следующего вида:

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

Таблица умножения 10×10 в виде двумерного массива.

```
1 ...  
2 int mult[10][10];  
3 for (i = 0; i < n; i++) {  
4     for (j = 0; j < n; j++) {  
5         mult[i][j] = (i + 1) * (j + 1); {  
6     }  
7 }  
8 ...
```

Инициализировать матрицу можно сразу же при объявлении:

```
1  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:

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

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

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

Решение

```
1  typedef int Matrix[100][100];
2  int main() {
3      Matrix matrix_a, matrix_b, matrix_c;
4      int i, j, k, size;
5      /* input */
6      for (i = 0; i < size; i++) {
7          for (j = 0; j < size; j++) {
8              c[i][j] = 0;
9              for (k = 0; k < size; k++) {
10                 c[i][j] += a[i][k] * b[k][j];
11             }
12         }
13     }
14     /* output */
15     return 0;
16 }
```


Самые полезные ссылки

Прогресс:

<https://docs.google.com>

Как идёт обучение у нас:

<http://mymath.info>

Как раньше шло обучение в Москве:

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