Практикум на ЭВМ
План, технические моменты и вспоминаем C

Баев А.Ж.

Казахстанский филиал МГУ

14 сентября 2019

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 в классе + 10 баллов дома

2 часть (C)

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. Кальулятор
2. Редактор изображений
3. Paint
4. Арканоид
5. Изметрическая 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

bash

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

```
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      if ((input_a > 0) && (input_b > 0)) {
7          sum = input_a + input_b;
8          printf("%d\n", sum);
9      } else {
10         puts("Bad␣input");
11     }
12     return 0;
13 }
```

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

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

Оформление кода

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

```
1 sudo apt install python3-pip
2 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
```

Компиляция

Обычный режим

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

Отладка

Для отладки

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

```
1 gdb 01
```

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

Makefile с компиляцией и проверкой кода

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 (colomns * 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 }
```

На дом

- Разобраться с gdb
- Прочитать про стиль
- Разобраться с cpplint
- Разобраться с Makefile