Задания по программированию студентам группы 8113 ММФ НГУ

Александр Геннадьевич Фенстер

2009 г.

Для получения зачёта по программированию необходимо и достаточно выполнить следующие действия:

1. На практических занятиях в терминальном классе сдать все задачи, перечисленные ниже. Сдача задач по электронной почте и другими «удалёнными» способами допускается лишь в виде исключения (болезнь и т. п.).

2. Посетить все семинары и написать все «пятиминутки» на оценку не ниже «±». В случае пропуска семинара необходимо попросить у преподавателя или взять на сайте http://8113.fenster.name задание с соответствующего семинара, выполнить его и сдать.

3. Сдать устный зачёт в конце семестра (семь задач по основным темам).

Перечисленные ниже задания должны быть сданы в виде исходного кода программы на языке C. Для компиляции программ можно использовать любой из доступных компиляторов. Для удобства список задач разделён на части, при этом задачи имеют сквозную нумерацию.

Попытки сдать чужое решение задачи не приветствуются. В случае появления сомнений в авторстве программы преподаватель может переформулировать задачу или попросить реализовать дополнительную функциональность.

Часть 1. Простые программы с циклами

Задание 1. Числа Фибоначчи. Вычислите первые N членов ряда Фибоначчи:

$$f_0 = 0,\ f_1 = 1,\ f_n = f_{n-1} + f_{n-2}.$$

Не используйте массивы и рекурсию.

Задание 2. Алгоритм Евклида. Для нахождения наибольшего общего делителя двух чисел удобно использовать следующий метод, называемый *алгоритмом Евклида*:

$$\text{НОД}\,(a, b) = \begin{cases} b, \text{ если } a \,\%\, b = 0; \\ \text{НОД}\,(b,\ a \,\%\, b) \text{ в противном случае,} \end{cases}$$

где $x \,\%\, y$ — остаток от деления x на y. Реализуйте вычисление НОД двух чисел по алгоритму Евклида. Не используйте рекурсию: рекурсивная реализация будет разобрана на семинаре.

Задание 3. Уравнение. Пусть функция $f : \mathbb{R} \to \mathbb{R}$ и известно, что $f(x)$ строго монотонна на отрезке $[a, b]$. Найдите приближённое решение уравнения $f(x) = 0$ с точностью до $\varepsilon = 10^{-3}$ на этом отрезке или сообщите, что решения нет. Для решения задачи определите в программе функцию `float f(float x)` и переменные `a` и `b`, например, так:

```
float a = 0;
float b = 2;

float f(float x)
{
    return (x * x - 1);
}
```

В этом примере уравнение имеет вид

$$\begin{cases} x^2 - 1 = 0, \\ x \in [0, 2]; \end{cases}$$

его решение: $x = 1$.

Не нужно делать перебор чисел от a до b с некоторым шагом. Т. к. функция монотонна, можно найти корень методом деления пополам.

Задание 4. Интеграл. В программе задана непрерывная на отрезке $[a, b]$ функция $f : \mathbb{R} \to \mathbb{R}$. Вычислите приближённое значение интеграла

$$\int_a^b f(x)\mathrm{d}x,$$

подсчитав площадь фигуры под графиком функции (разбейте эту фигуру на прямоугольники или трапеции с высотой $\varepsilon = 10^{-3}$).

Часть 2. Работа с массивами

Здесь и далее фразы типа «дан массив ...» нужно понимать как «с клавиатуры или из файла вводится массив ...». Для ввода массива необходимо использовать цикл, читающий последовательно каждый из его элементов.

Задание 5. Проверка свойств элементов. Дан одномерный массив натуральных чисел размера N. Проверьте, что все его элементы являются простыми числами. Код, проверяющий число на простоту, вынесите в отдельную функцию `int isprime(int a)`. Ясно, что если нашлось хотя бы одно составное число, то дальнейшие проверки производить не нужно.

Задание 6. Умножение матриц. Даны две матрицы A и B размера $N \times N$. Выведите на экран элементы матрицы $C = A \cdot B$:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}.$$

Частая ошибка: многие забывают инициализировать матрицу C нулями.

Задание 7. Бинарный поиск. Дан массив, про который известно, что его элементы упорядочены в порядке неубывания. Введите с клавиатуры несколько чисел и, используя метод деления пополам, определите, встречаются ли эти числа в массиве. Не используйте рекурсию: рекурсивная реализация будет разобрана на семинаре.

Бинарный поиск является, пожалуй, лучшим способом поиска элемента в упорядоченном массиве. В наихудшем случае для поиска элемента (или определения того, что число в массиве отсутствует) необходимо сделать всего $[\log_2 N] + 1$ сравнений.

Часть 3. Работа со строками

В задачах из этой части можно считать, что длина всех строк ограничена (например, меньше 1000 символов).

Задание 8. Палиндром. Проверьте, что данная строка является палиндромом, т.е. читается одинаково слева направо и справа налево (без учёта пробелов и знаков препинания). **Не используйте дополнительную строку и не удаляйте пробелы при чтении исходной строки.** Используйте для чтения строки функцию `fgets`. Вот пример строки, являющейся палиндромом:

```
Он дивен, палиндром, и ни морд, ни лап не видно...
```

Задание 9. Системы счисления. В системе счисления с основанием b для записи чисел используются цифры 0, 1, ..., $b-1$ (если $b > 10$, используют буквы: $A = 10$, $B = 11$ и т. д.). Значение числа $\overline{a_n a_{n-1} \ldots a_1 a_0}_b$ (a_i — цифры числа, индекс $_b$ показывает основание системы) вычисляется по следующей формуле:

$$\overline{a_n a_{n-1} \ldots a_1 a_0}_b = \sum_{i=0}^{n} a_i b^i.$$

Например, значением числа 14_6 является $4 \cdot 6^0 + 1 \cdot 6^1 = 10$.

Введите с клавиатуры натуральное число b ($b > 1$) и строку, содержащую запись некоторого числа в системе счисления с основанием b. Вычислите значение этого числа в десятичной системе счисления.

Задание 10. Имя пользователя. Информация о пользователях хранится в системе Linux в текстовом файле `/etc/passwd`. Каждая строка этого файла хранит данные об одном пользователе. Вот пример такой строки:

```
fenster:x:1000:1000:Alexander Fenster,Teacher,,:/home/fenster:/bin/bash
```

Легко заметить, что в этой строке хранится несколько полей, разделённых двоеточием. Четвёртое (считая с нуля) поле в свою очередь хранит список полей, разделённых запятыми. Напишите программу, которая запрашивает с клавиатуры логин и определяет имя этого пользователя по файлу `/etc/passwd`.

Часть 4. Алгоритмы сортировки

В двух следующих заданиях программа должна брать входные данные из файла `input.txt`. В первой строке этого файла записано число N — количество чисел, которые необходимо отсортировать (известно, что $1 \leqslant N \leqslant 100000$). Далее в файле записаны N чисел типа `int`. В файл `output.txt` необходимо выдать эти N чисел в порядке неубывания.
Пример файла `input.txt`:

```
5
3 1 5 4 2
```

Пример файла `output.txt`:

```
1 2 3 4 5
```

Задание 11. Быстрая сортировка (обменная сортировка с разделением). Реализуйте алгоритм быстрой сортировки массива.

Задание 12. Пирамидальная сортировка (сортировка при помощи кучи). Реализуйте алгоритм пирамидальной сортировки массива.

Описание этих алгоритмов находится в отдельном файле.

Часть 5. Работа со списками

Задание 13. Сортировка вставкой в список. Отсортируйте последовательность чисел из файла путём вставки их в упорядоченный список (список изначально пуст). Длина последовательности заранее неизвестна.
Пример чтения чисел из файла до конца файла:

```
FILE *f = fopen("input.txt", "r");
int a;
while (fscanf(f, "%d", &a) == 1) {
        /* do something */
}
fclose(f);
```

Функция `fscanf` в нормальном случае возвращает количество прочитанных аргументов. Если она вернула не 1, делается вывод о том, что либо достигнут конец файла, либо произошла ошибка (например, в файле записано не число), и цикл завершается.
Пример описания списка:

```
struct item {
        int data;
        struct item *next;
};
struct item *head = 0;
```

Задание 14. Количество вхождений слов в текст. Подсчитайте, сколько раз каждое слово встречается в тексте. Текст читается из файла `input.txt`, для простоты можно считать, что словом является любая последовательность латинских букв, а всё, что не является латинской буквой, считается разделителем. Максимальная длина слова равна 10 символам.

Для хранения информации о количестве вхождений каждого слова можно определить следующую структуру:

```
struct item {
        char word[11];
        int count;
        struct item *next;
};
```

Часть 6. Индивидуальное задание на графы

Задание 15. Индивидуальное задание. Список заданий находится в отдельном файле.