Задания по программированию студентам группы 8205 ФИТ НГУ. Первый семестр

Александр Геннадьевич Фенстер

2008 г.

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

1. На практических занятиях в терминальном классе сдать как минимум пятнадцать из двадцати задач, перечисленных 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 или получения оценки «−» необходимо попросить у преподавателя или взять на сайте http://8205.fenster.name задание с соответствующего семинара, выполнить его и сдать.

3. Сдать устный зачёт в конце семестра (некоторые студенты могут быть от него освобождены).

Зачёт будет дифференцированным, т. е. за него будет выставляться оценка («отлично», «хорошо», «удовлетворительно»). На оценку влияет количество сданных задач и результат устного зачёта.

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

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 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: $x = 1$.

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

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

Задание можно переформулировать следующим образом: вычислить с некоторой точностью площадь фигуры, ограниченной графиком функции $y = f(x)$ и прямыми $x = a$, $x = b$, $y = 0$.

Часть 2. Работа с массивами

Здесь и далее фразы типа «дан массив ...», «дана строка ...» и т. п. нужно понимать как «из файла `input.txt` вводится массив, строка, ...». 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:

а) все его элементы являются простыми числами;

б) среди его элементов есть хотя бы одно простое число.

Код, проверяющий число на простоту, вынесите в отдельную функцию `int isprime(int a)`.

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

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

(строки матрицы A умножаются на столбцы матрицы B).

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 очень хорошим 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

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.** Вот пример строки, являющейся палиндромом:

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

Задание 9. Системы счисления — 1. 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 $\overline{a_n a_{n-1} \dots a_1 a_0}_b$ (a_i — цифры числа, индекс $_b$ показывает основание системы) вычисляется по следующей формуле:

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

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

Введите с клавиатуры натуральное число b ($2 \leqslant b \leqslant 16$) и строку, содержащую запись некоторого числа в системе счисления с основанием b. Вычислите значение этого числа в десятичной системе счисления.

Задание 10. Системы счисления — 2. Вычислите, как будет записываться введённое с клавиатуры десятичное натуральное число в системе счисления с основанием b ($2 \leqslant b \leqslant 16$). Для получения результата необходимо разделить данное число на b. Остаток от деления будет являться последней цифрой результата (помните, что для систем с основанием, большим десяти, необходимо использовать буквы A, B, ..., F). Частное необходимо снова разделить на b, остаток будет являться предпоследней цифрой результата, и так далее. Деление повторять до

тех пор, пока частное не будет равняться нулю, остаток в этом случае будет являться первой цифрой результата.

Для получения символа, соответствующего данному числу ($0 \to$ `'0'`, $1 \to$ `'1'`, ..., $15 \to$ `'F'`), не следует писать много условных конструкций. Эта операция легко делается путём добавления некоторого числа к символу `'0'` или `'A'`.

Задание 11. 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`.

Задание 12. Разбор формата CSV. CSV (comma-separated values) — популярный (но не самый удобный) формат хранения табличных данных. Файл в формате CSV состоит из строк, в каждой из которых хранятся данные из нескольких ячеек, разделённые запятыми, например, так:

```
Фамилия,Имя,Отчество
Иванов,Иван,Петрович
Петров,Василий,Сергеевич
```

Проблемы возникают, если ячейка таблицы содержит запятую или перевод строки. В таком случае содержимое ячейки заключается в двойные кавычки. Если же ячейка содержит двойную кавычку, эта кавычка дублируется. В следующем примере приведена таблица из трёх строк и трёх столбцов (во второй строке третья ячейка пустая):

```
следующая ячейка содержит перевод строки,"до перевода строки
после перевода строки",а это третий столбец
следующая ячейка содержит кавычку и запятую,"вот они: "","
в этой строке вторая и третья ячейка содержат по одной кавычке","""","""
```

При форматировании таблицы в языке разметки HTML таблица начинается тегом `<table>` и заканчивается тегом `</table>`. Строка таблицы начинается тегом `<tr>` и заканчивается тегом `</tr>` (table row). Ячейка таблицы начинается тегом `<td>` и заканчивается тегом `</td>` (table detail). Перевод строки обозначается тегом `<br/>` (break).

Прочитайте из файла таблицу в формате CSV и выведите в файл таблицу в формате HTML. Для простоты считайте, что таблица не содержит символов < и >.

Если бы эти символы допускались, необходимо было бы заменять < на последовательность `&lt;` (less than), а > на `&gt;` (greater than). Символ `&`, который может встретиться в тексте, при этом необходимо заменять на `&amp;`.

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 $1 \leqslant N \leqslant 10000$). 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 13. Простая сортировка. Реализуйте любой простой алгоритм сортировки (простой выбор, простые вставки, пузырёк, ...).

Задание 14. Быстрая сортировка (обменная сортировка с разделением). Реализуйте алгоритм быстрой сортировки массива.

Задание 15. Пирамидальная сортировка (сортировка при помощи кучи). Реализуйте алгоритм пирамидальной сортировки массива.

С подробным описанием алгоритмов быстрой и пирамидальной сортировки можно ознакомиться в отдельном файле.

Часть 5. Динамические структуры данных

Задание 16. 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 = NULL;
```

Задание 17. 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;
};
```

Задание 18. Хэш-таблицы. *Хэш-функцией* называется функция

$$f : K \to \{0, \ldots, N-1\},\ N \in \mathbb{N},$$

сопоставляющая каждому элементу *множества ключей* K натуральное число от 0 до $N-1$. В данном примере множеством K является множество всех слов из букв латинского алфавита.

При помощи хэш-функции каждому слову сопоставляется хэш-код, по которому определяется позиция в массиве из N элементов. Ситуация, при которой два разных слова имеют один и тот же код, называется *коллизией*.

Хэш-таблицей называется массив из N элементов, в i-м элементе которого хранится указатель на начало списка слов, хэш-код которых равен i.

Подсчитайте количество вхождений слов в текст, используя для хранения слов хэш-таблицу.

Задание 19. Сортировка вставкой в дерево. Отсортируйте последовательность чисел из файла путём вставки их в двоичное дерево поиска (дерево изначально пусто). Длина последовательности заранее неизвестна.
Пример описания дерева:

```
struct item {
        int data;
        struct item *left, *right;
};
struct item *root = NULL;
```

Задание 20. Обход в глубину. Прочитайте из файла список рёбер неориентированного графа и выведите номера вершин, доступных из вершины №1, используя обход в глубину. Считайте, что граф содержит не более 100 вершин (можно использовать матрицу смежности заданного заранее размера).