Структуры; динамические списки

А. Г. Фенстер, fenster@fenster.name

9 апреля 2010 г.

Конспект семинара №9 по программированию для студентов 1 курса ММФ НГУ. Помните, что чтение конспекта не делает посещение соответствующего семинара необязательным! О найденных ошибках и неточностях, пожалуйста, пишите мне по адресу fenster@fenster.name. Также буду рад получить любые комментарии по поводу этого текста.

Цель этого семинара — познакомить вас с понятием динамического списка (он же связный список). Такие списки применяются, когда размер данных, используемых программой, сложно предсказать заранее и при этом нет необходимости размещать все данные в одном блоке памяти подряд (как это происходит при использовании массивов).

1 Структуры

Структурой называется определяемый пользователем тип данных, который может хранить несколько значений (*полей*) различных типов. По сути, структура — объединение нескольких (возможно, разнотипных) переменных в одной для удобства их совместного использования.

В следующем примере определяется тип `struct human`, который хранит информацию об одном человеке:

```
struct human
{
    char *lastname, *firstname; /* фамилия, имя */
    int year;                   /* год рождения */
};
```

Слово `struct` является зарезервированным. После такого определения мы можем создать переменную типа `struct human` и вообще использовать `struct human` просто как ещё один тип данных:

```
struct human h1;          /* переменная типа struct human */
struct human employee[N]; /* массив структур */
struct human *ph = &h1;   /* указатель на структуру */
```

Для обращения к конкретному члену (полю) структуры используется оператор `.` («точка»).

```
h1.year = 1983;
```

Обращение к полю струкруры через указатель на неё — настолько частая операция, что для неё выделен особый оператор `->` («стрелка»). Следующие две строки эквивалентны:

```
(*ph).year = 1983;
ph->year = 1983;   /* следует использовать этот вариант */
```

«Точка» используется для обращения к полю структуры; «стрелка» используется для обращения к полю структуры через указатель, хранящий адрес этой структуры.

«Официальные» названия операторов `.` и `->` — *селекторы членов структуры.*

2 Наивный подсчёт количества слов

Задача. Используя функцию `readword` с предыдущего семинара, подсчитайте количество вхождений каждого слова в файл `input.txt`. Считайте, что количество различных слов не превосходит `N` (тогда для хранения можно использовать обычный массив заданного заранее размера) — именно поэтому использовано слово «наивный».

Храните слова в структуре примерно такого вида:

```
struct word
{
    char *word;
    int count;
};
```

Не забывайте, что функция `readword` выделяет новую память под каждое прочитанное слово. Прочитав слово, необходимо разобрать два случая: либо это «новое», не встречавшееся ране слово (добавляем его в конец массива с количеством вхождений, равным единице), либо же это слово уже встречалось ранее (увеличиваем счётчик у найденного при помощи цикла с `strcmp` слова **и удаляем память, выделенную функцией** `readword`, потому что нам не нужно запоминать это же слово второй (третий, четвёртый, ...) раз.

Задача. Как изменится реализация предыдущей задачи, если количество различных слов не ограничено?

3 Динамические списки

Начальная информация о списках в отдельном файле.

4 Проверочное задание («пятиминутка»)

Тема: простейшая работа со списками (см. задание).