1. Уточнение задания

Прежде всего уточним требования к программе.

Взаимодействие с пользователем. После запуска программы пользователь может вводить сколько угодно запросов. После каждого из них программа выводит найденную информацию, а затем ожидает следующего запроса. Работа заканчивается, если пользователь дал «пустой» запрос (нажал Enter без ввода символов).

Синтаксис запросов. а) Допустимы ли пробелы? б) Регистр в хим. символах.

Принимаем следующие решения:

а) Пробелы допускаются. Для упрощения разбора программа должна предварительно удалить все пробелы из строки запроса.

б) Программа различает регистр букв в химических символах. В двухбуквенных символах первая буква заглавная, а вторая — строчная.

Реакция на ошибки. Любая программа должна учитывать «человеческий фактор», т.е. возможность некорректных действий пользователя. В рассматриваемой задаче возможны ошибки в записи запроса. Главное требование — что бы ни ввел пользователь, программа не должна ломаться, т.е. аварийно завершать работу. На ошибки следует реагировать диагностическим сообщением и переходить к вводу нового запроса.

2. Общая схема программы

1. Загрузить из файла таблицу изотопов.
2. Получить очередной запрос от пользователя.
3. Если запрос пустой, закончить работу.
4. Выполнить разбор запроса, а именно:
   – определить его тип (1, 2, 3, 4; 0 – ошибка),
   – выделить параметры — символ элемента и массовое число.
5. В зависимости от типа запроса выполнить поиск информации и вывести результат.
6. Перейти к п. 2.

Эта схема реализована в функции main:

```
int main (void)
{
   char str[64];   // строка запроса пользователя
   int massnumber; // массовое число из запроса

   loaddata();     // загрузка таблицы изотопов из файла

   for (;;)        // бесконечный цикл ввода запросов
   {
      printf("? ");
      fgets(str, 64, stdin);      // ввод строки запроса
      if (!delspaces(str)) break; // если пусто, закончить работу

      switch (query(str, &massnumber))
      {
         case 1:
            search1(str);
            break;
         case 2:
            search2(str);
            break;
```

```
            case 3:
               search3(str, massnumber);
               break;
            case 4:
               search4(massnumber);
               break;
            default:
               printf("Ошибка в запросе!\n");
         }
      } // конец цикла запросов

      return 0;
   }
```

Функция `main` обращается к следующим функциям:

`void loaddata(void)` — загружает таблицу изотопов из файла в массив.

`int delspaces(char *str)` — убирает все пробелы из строки `str`, возвращает длину получившейся строки.

`int query(char *str, int *massnumber)` — выполняет разбор запроса в строке `str`, возвращает тип запроса (1, 2, 3, 4 или 0). После работы в `str` находится символ элемента, в переменную `massnumber` записан атомный номер (в случае запросов 3 или 4).

`search1(char *str)` — поиск по запросу типа 1 и вывод найденной информации.

`search2(char *str)` — поиск по запросу типа 2 и вывод найденной информации.

`search3(char *str, int massnumber)` — поиск по запросу типа 3 и вывод найденной информации.

`search4(int massnumber)` — поиск по запросу типа 4 и вывод найденной информации.

Вывод информации о найденных изотопах во всех случаях имеет один и тот же формат. Чтобы не дублировать одинаковые обращения к `printf` в четырех функциях поиска, можно оформить вывод в виде самостоятельной функции `printdata` и вызывать ее из каждой функции `search`.

3. Хранение базы данных и доступ к ней функций

База данных используется функциями `loaddata`, `search1` – `search4` и `printdata`. Чтобы все эти функции имели к ней доступ, ее следует разместить во внешних переменных:

```
struct isotope_data
{
   int    z, a;
   char   el[3];
   double mass, abund;
} base[300];            // таблица изотопов – массив структур

static int N = 0;       // размер таблицы (определяется при загрузке)
```

4. Разбор запроса

Согласно условию задачи, возможны запросы 4-х типов:

El — найти все изотопы элемента *El*,

El[*A*] — найти изотоп элемента *El* с массовым числом *A*,

El[] — найти наиболее распространенный изотоп элемента *El*,

∗[*A*] — найти изотопы любых элементов с массовым числом *A*,

где *El* — символ химического элемента (строка, состоящая из одной или двух букв), *A* — массовое число изотопа (целое десятичное число).

Определить тип запроса можно, проверив три признака:

1. Первый символ ‘∗’? Если да, то это запрос 4-го типа.
2. Присутствуют ли квадратные скобки? Если нет, то запрос 1-го типа.
3. Присутствует ли число внутри скобок? Если да, то это запрос 2-го или 4-го типов, в противном случае — запрос 3-го типа.

Не все сочетания этих признаков являются допустимыми. Если первый символ ∗, то обязательны квадратные скобки, причем эти скобки не могут быть пустыми.

Для последующего поиска в базе данных из запроса необходимо извлечь (выделить) символ элемента (в виде строки) и массовое число (в виде значения типа `int`).

В процессе разбора запроса и поиска в базе данных могут быть полезны следующие стандартные библиотечные функции для работы со строками и символами:

`isspace(`*c*`)` — возвращает ненулевое значение (логическое *истина*), если символ *c* является пробельным (т. е. пробелом, символом табуляции, символом перехода на новую строку и т. п.).

`strlen(`*s*`)` — возвращает длину строки *s*.

`strchr(`*s*`,` *c*`)` — ищет символ *c* в строке *s*. Возвращает указатель на найденный символ либо `NULL`, если символа *c* в строке нет.

`strcmp(`*s1*`,` *s2*`)` — сравнивает строки *s1* и *s2*, возвращает целое число 0, если строки совпадают.

`atoi(`*s*`)` — преобразует десятичное целое число, записанное в виде строки цифр *s*, в значение типа `int`.

(Для использования этих функций необходимо включить в программу через `#include` стандартные заголовочные файлы `ctype.h`, `string.h` и `stdlib.h`).

Алгоритм разбора в функции `query(char *str, int *massnumber)` может быть таким:

1. Ищем в строке запроса `str` открывающую скобку `[`. Если она найдена, заменяем ее признаком конца строки (`'\0'`), чтобы сделать символ элемента самостоятельной строкой, и запоминаем указатель `p` на следующий символ. Если же скобка не найдена, то разбор закончен — это запрос типа 1.

2. В строке с началом `p` ищем закрывающую скобку `]`. Если она найдена, заменяем ее признаком конца строки (`'\0'`), чтобы выделить массовое число в виде отдельной строки. Если же скобка не найдена, то в запросе ошибка.

3. Проверяем длину строки с началом `p`. Если она равна нулю (массовое число отсутствует), то разбор закончен — это запрос типа 3.

4. С помощью функции `atoi(p)` извлекаем массовое число и сохраняем его в переменной `massnumber`.

5. Проверяем первый символ в `str`. Если это ∗, то имеем запрос типа 4, в противном случае — запрос типа 2. Разбор закончен.

В результате разбора `str` содержит символ элемента, а в переменной `massnumber` в случае запросов типа 2 или 4 находится массовое число.

В приведенном выше упрощенном алгоритме разбора отсутствует контроль ошибок (за исключением проверки парности квадратных скобок). Например, в запросах типов 1 и 3 следовало бы убедиться, что первый символ в `str` отличен от `*`. Кроме того, желательно убедиться в выполнении следующих условий:

- длина химического символа равна 1 или 2;
- химический символ состоит только из букв, причем первая буква всегда заглавная, а вторая (если она есть) — строчная;
- символ `*` должен быть единственным символом перед скобкой;
- после закрывающей скобки нет никаких символов;
- внутри скобок содержатся только цифры;
- значение массового числа находится в интервале от 1 до 295 (или даже от 1 до 240).

Подобные проверки позволят выдать пользователю подробное сообщение с указанием характера допущенных ошибок. Однако отсутствие проверок не должно привести к серьезным нарушениям работы программы в случае ошибочного запроса — просто не будет найдена требуемая информация по причине несуществующего химического символа или отсутствующего в базе массового числа. Кстати, проверку предпоследнего условия (только цифры внутри скобок) легко обеспечить, если вместо функции `atoi` использовать похожую функцию `strtol` с расширенными возможностями (см. ее описание в Приложении Б [раздел Б.5] в книге Кернигана и Ритчи).