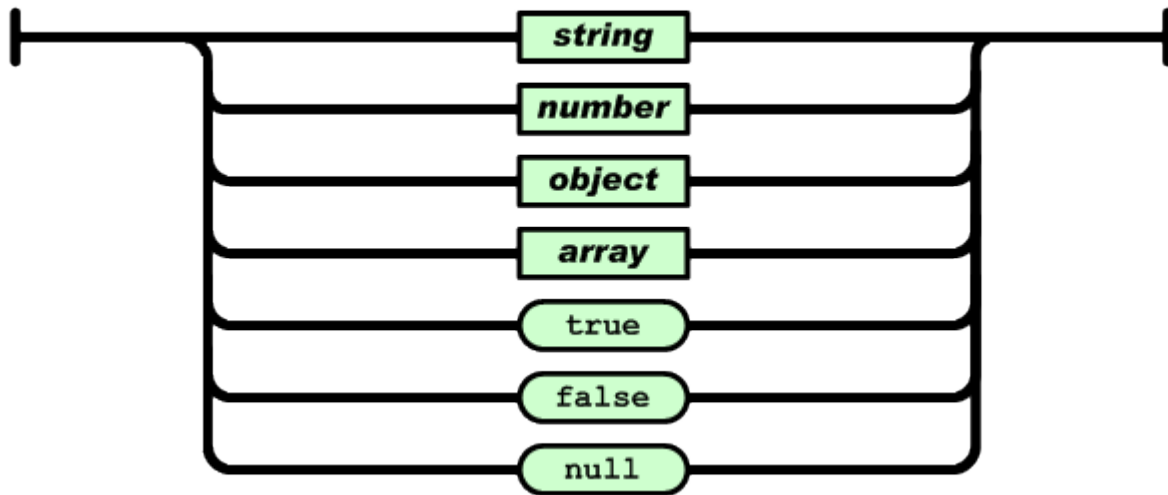


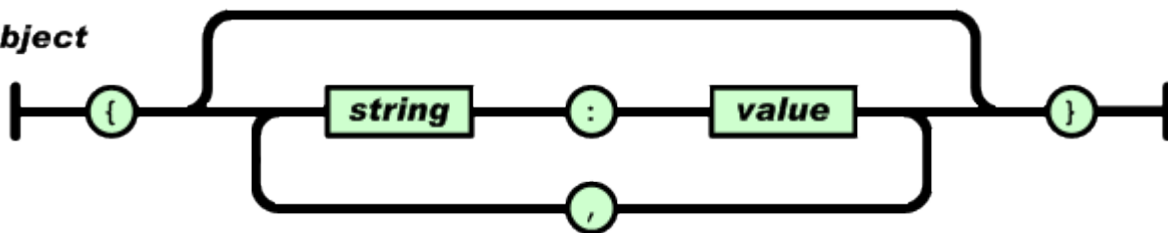
Практическое занятие «Освоение отладчика gdb»

В качестве рабочего материала предлагается библиотека cJSON, предназначенная для **генерации** и **разбора** данных в формате **JSON**. Описание формата JSON:

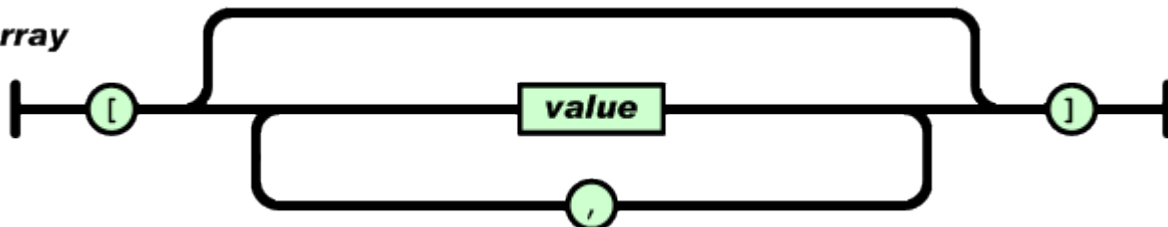
value

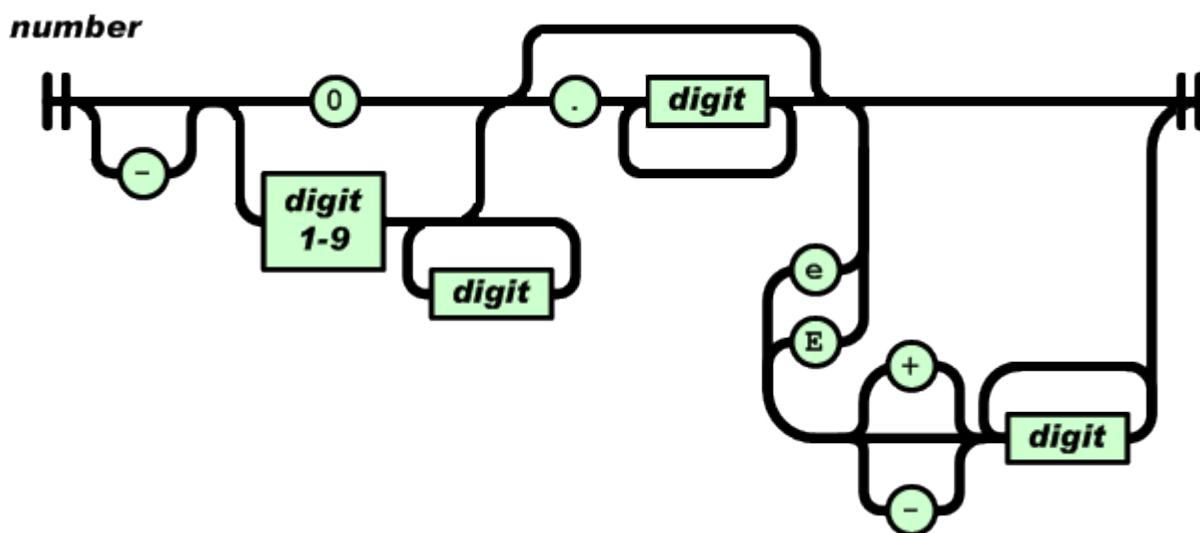
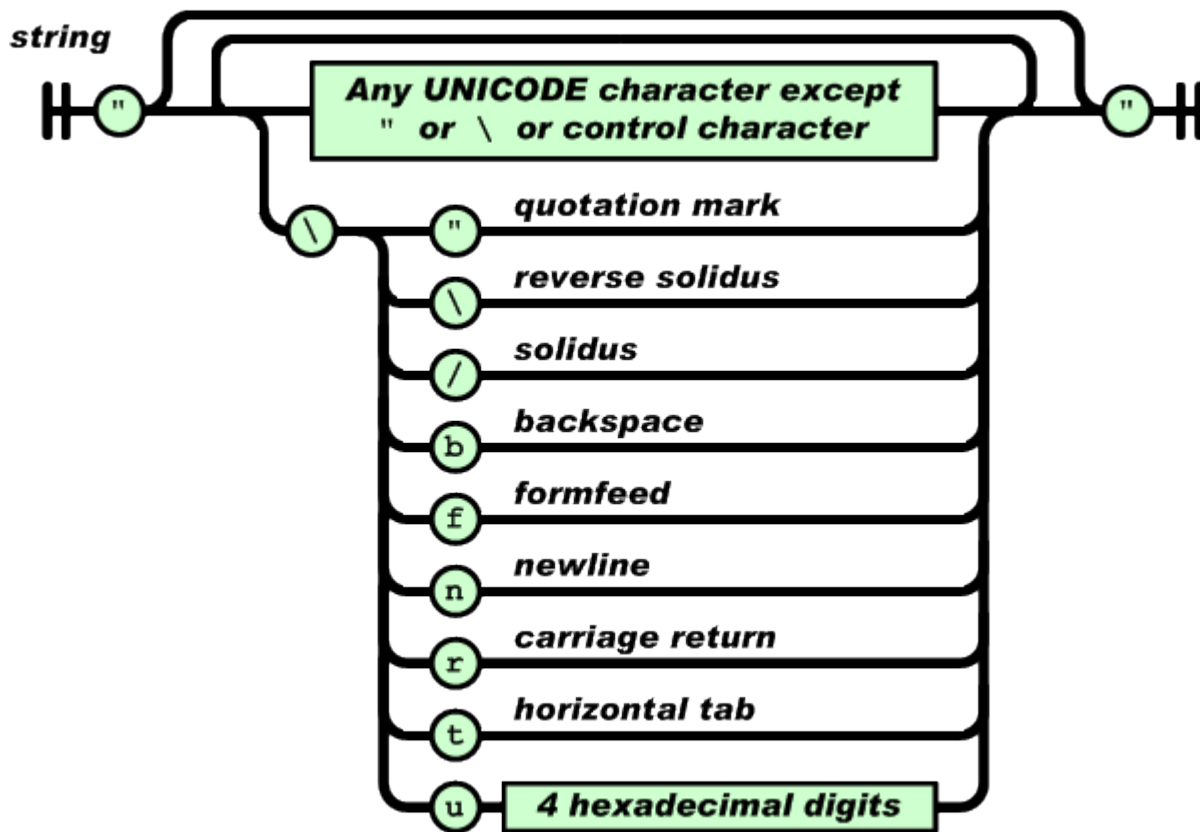


object



array





В реализацию функций **разбора** JSON умышленно добавлены ошибки.

Часть 1 «Освоение интерактивной отладки».

Дано

- исходный текст сценария: `task1.c`

Требуется

Скомпилировать сценарий с отладочной информацией, — ключ `-g`:

```
gcc -g task1.c cJSON.c -o task1 -lm
```

с помощью `gdb` выполнять ряд шагов по «передвижению» по отлаживаемому коду для получения значений переменных:

1. Значение `value[2]` при 1-м вызове функции `parse_object`.
2. Значение `item->type` (число) перед возвратом из 2-го по счету вызова функции `parse_value`.
3. Значение переменной `len` функции `parse_string` на момент выполнения строки
`out = (char *) cJSON_malloc (len + 1);` при 4-м вызове функции.

Контроль

- Отправка ответов на вопросы в `ejudge`.

Справочник по `gdb`

`gdb`

Компиляция с отладочной информацией:

```
gcc -g prog.c -o prog
```

Запуск отладчика

```
gdb prog
```

Команды отладчика

quit – завершение работы отладчика.

run – запуск программы на выполнение. Команда доступна только, если программа еще не запущена. При подключении к работающему процессу или исследовании `core`-файлов данная команда не доступна. Допускается указывать аргументы командной строки и перенаправление потоков ввода-вывода, как если бы программа запускалась командным процессором. Пример:

```
(gdb) run < 001.in > 001.out
```

Здесь программа запускается на выполнение с файлом `001.in`, перенаправленным на стандартный поток ввода, и файлом `001.out`, в который перенаправляется вывод на стандартный поток вывода.

Выполнение программы может быть прервано в любой момент нажатием на комбинацию *Ctrl-C*. Отладчик приостановит выполнение программы и выдаст приглашение ко вводу очередной команды отладчика.

Выполнение программы также приостанавливается при получении программой любого сигнала, при завершении программы либо достижении точки останова.

bt (или *backtrace*) – распечатать стек вызовов. Стек вызовов печатается от самой вложенной функции к функции `main`. Для каждого стекового фрейма печатается адрес в коде точки вызова, название функции и параметры, переданные в функцию, а также позиция в исходном коде. Команда *bt full* дополнительно печатает значения локальных переменных.

up – переход на указанное количество фреймов вверх по стеку вызовов функций. Если аргумент `y` команды не указан, он принимается равным 1 (переход на один фрейм вверх, то есть переход к функции, которая вызвала текущую функцию).

down – переход на указанное количество фреймов вниз по стеку вызовов функций. Если аргумент `y` команды не указан, он принимается равным 1 (переход на один фрейм вниз, то есть переход к функции, которая была вызвана в текущей точке текущей функции).

info frame – получить информацию о текущем стековом фрейме.

info locals – получить информацию о значениях локальных переменных текущего стекового фрейма.

p (или *print*) – напечатать значение выражения. Аргументом команды может быть почти произвольное выражение языка Си, даже включающее в себя вызовы функций программы, если, конечно, отлаживаемый процесс существует. Таким образом, вызовы функций недоступны при «посмертной» отладке. Если аргумент команды не указан, берется аргумент, который был указан в команде *p* в последний раз.

l (или *list*) – напечатать исходный код. Команду можно использовать во многих вариантах, часть из которых перечислена ниже.

(gdb) l	<i>напечатать очередные 10 строк исходного файла</i>
(gdb) l -	<i>напечатать предыдущие 10 строк исходного файла</i>
(gdb) l 200	<i>напечатать 10 строк в окрестности 200 строки текущего файла</i>
(gdb) l prog.c:200	<i>напечатать 10 строк в окрестности 200 строки файла prog.c</i>
(gdb) l main	<i>напечатать 10 строк в окрестности начала функции main</i>
(gdb) l *0x0806e502	<i>напечатать 10 строк в окрестности кода по указанному адресу</i>

b (или *break*) – установка точки останова. Параметром команды является точка в программе, помечаемая как точка останова. Команду можно использовать во многих вариантах, часть из которых перечислена ниже.

(gdb) b main	<i>установить точку останова в начале функции main</i>
(gdb) b 200	<i>установить точку останова на 200 строке текущего файла</i>
(gdb) b prog.c:200	<i>установить точку останова на 200 строке файла prog.c</i>
(gdb) b *0x0806e502	<i>установить точку останова по указанному адресу</i>

c (или *continue*) – продолжить выполнение программы.

finish — продолжить выполнение программы до достижения конца текущей функции

n (или *next*) — сделать указанное количество шагов выполнения (по умолчанию 1). Вызовы функций рассматриваются как одна инструкция, то есть вызванные функции выполняются в обычном, а не пошаговом режиме.

s (или *step*) – сделать указанное количество шагов выполнения программы (по умолчанию 1). При вызовах функций выполнение входит в функции в пошаговом режиме.

«Посмертная» отладка

Снятие ограничение на размер файлов дампа памяти

```
ulimit -c unlimited
```

Запуск отладки по файлу дампа памяти:

```
$ gdb ./prog core.7911
```