A JSON parser and formatter using PEGPP

Jsonparser reads a JSON from standard input, parses it with PEGPP and builds a syntax tree. The type of each node in the tree is a C++ variant holding the C++ type chosen to represent the corresponding JSON value type. The mapping of JSON types to C++ is defined in json.h:

```
struct json_type;

using null_type = std::nullptr_t;
using bool_type = bool;
using number_type = double;
using string_type = std::string;
using array_type = std::vector<json_type>;
using object_type = std::map<string_type, json_type>;

using variant_type = std::variant<null_type, bool_type, number_type, string_type, array_type, object_type>;

struct json_type : variant_type
{
    using variant_type::variant_type;
};
```

The official syntax of JSON was taken from www.json.org, where it appears in the form of syntax diagrams. This was translated into an equivalent PEG grammar.

Once the syntax tree is built, a formatter visits the tree and prints it. This is the public interface of the formatter class:

```
class json_formatter
{
public:
    json_formatter(unsigned tabsize = 0, unsigned tabs = 0);
    std::string format(const variant_type &v);
};
```

If tabsize == 0 formatting is done in a compact way, without indentation or newlines. If tabsize != 0 formatting uses newlines and indentation with the given tab size. If tabs != 0, the whole print is further indented by the given number of tabs.

For the jsonparser program, the usage is:

```
jsonparser [tabsize = 4]
```

where the maximum value of tabsize is 16. Greater values are clipped to 16.

This parser uses error reporting. The file example.json contains a sample JSON and if used like this:

```
./jsonparser < example.txt
```

jsonparser will print an exact replica of the file. You may try other inputs and/or introduce errors in example.json to check the error messages reported.