YAY - a LALR parser generator.

Syntax:

```
yay sourcefile Parser=outfile [option]*

(+|-)LR2 (-)               (+|-)Verbose (-)
(+|-)Warnings (+)          Description=file
Header=file                INSTallation=file
Language=C|C++ (C)         Parser=file
```

Examples:

```
yay cgram.y parse=myparse.c
c myparse.c

yay ccgram.y lang=c++ pars=myparse.cpp
```

Options:

`sourcefile`

is a file containing YAY input.

`Language=C`

produces parsing tables in the C programming language. This is the default.

`Language=C++`

produces parsing tables in the C++ programming language. Note that YAY only produces the tables; the routines that use the tables to parse input are predefined.

`+LR2`

says that the YAY input describes an LALR(2) grammar. Without `+LR2`, YAY assumes that the grammar is LALR(1). The manual describes modifications that need to be made for LALR(2) grammars.

`Description=file`

translates the parsing tables into a format that humans can read, and writes this output into the given file.

`Header=file`

writes token definitions and other information necessary for separate compilation, to the named file.

`INSTallation=file`

tells YAY where to find the installation file. The installation file tells where various software components have been installed. For more information, see the section on *Installation Files* below.

If you do not specify an `INSTallation=` option on the command line, YAY checks for an environment variable named `YAY_INST` and uses its value as the name of the installation file. If this environment variable does not exist, YAY uses the default installation file.

`Parser=file`

writes the resulting source code for the parser into the named file. If this option is omitted, YAY just checks the syntax of your input.

`+Verbose`

produces verbose output -- everything that can be flagged is flagged.

`-Warnings`

suppresses a number of warning messages that YAY normally issues.

Description:

YAY converts your context-free grammar into a C or C++ program that is written to the file specified by the `Parser=` option.

If you use the `Description=` option, YAY writes a full description of the grammar to the specified file. YAY only displays a brief message on the standard output, summarizing conflicts (and other information if you specify `+Verbose`). On the other hand, if you do not use the `Description=` option, YAY writes more information to standard output, including descriptions of the states where conflicts occur. In this case, YAY actually provides additional information to help you identify the source of the conflicts; if you ask for a description file, YAY outputs less information when reporting the conflicts because it assumes you can track down additional information by looking at the description file. For this reason, you can sometimes get a quicker idea of what has gone wrong if you do not ask for a description file.

C++ Parsers

In general, you only need to use `Language=C++` if you intend `YYSTYPE` to contain a C++ object with constructors. If you intend to compile the parser with C++ but the `%union` statement does not have any elements that need constructors, it's best to use `Language=C` to get more efficient C code.

If `YYSTYPE` does contain elements that need constructors, you need to define an appropriate constructor-like function for the `YYSTYPE` type. This function should have the prototype

```
void name(YYSTYPE *p)
```

where `"name"` can be any valid name. In the declarations section of the grammar, you must then add the statement

```
#define YYSTYPE_INIT name
```

where `"name"` is the name of the constructor-like function.

With `Language=C++`, the `%union` statement generates a structure type rather than a union, since C++ does not allow objects with constructors to belong to unions.

In many cases, the same grammar may be processed with either `Language=C` or `Language=C++`.

Installation Files:

An installation file specifies the pathnames for software and data files used by YAY. Installation files are text files made up of comment lines and option lines.

Comment lines:

> Any line whose first non-blank character is `#` will be taken as a comment. Blank lines are also considered comments.

Option lines:

> Option lines have the format
>
> ```
> Keyword=pathname
> ```
>
> In this documentation, keywords are written with some letters in upper case and some in lower case. You may abbreviate keywords by omitting any or all of the letters shown in lower case. The remaining letters may be entered in either upper or lower case; the documentation simply uses upper case to show which characters may not be omitted.

In this version of YAY, there is only one valid option line:

```
Library=pathname
```

The pathname should be the directory containing the YAY parser template files (e.g. `yyparse.c`).

Notes:

If you define `YYALLOC`, the parser allocates its state and value stacks dynamically via malloc and free. This shrinks your parser and helps prevent stack overflows.