

## A simple tiger compiler

Generated by Doxygen 1.9.8



<b>1 Topic Index</b>	<b>1</b>
1.1 Topics . . . . .	1
<b>2 Struct Index</b>	<b>3</b>
2.1 Struct List . . . . .	3
<b>3 File Index</b>	<b>5</b>
3.1 File List . . . . .	5
<b>4 Topic Documentation</b>	<b>7</b>
4.1 SLP_interpreter . . . . .	7
4.1.1 Detailed Description . . . . .	7
4.1.2 Function Documentation . . . . .	7
4.1.2.1 max() . . . . .	7
4.1.3 <tt>__main__</tt> file . . . . .	7
4.2 Utils . . . . .	7
<b>5 Struct Documentation</b>	<b>9</b>
5.1 A_exp_ Struct Reference . . . . .	9
5.1.1 Detailed Description . . . . .	9
5.1.2 Member Enumeration Documentation . . . . .	10
5.1.2.1 anonymous enum . . . . .	10
5.1.3 Field Documentation . . . . .	10
5.1.3.1 [struct] . . . . .	10
5.1.3.2 exp . . . . .	10
5.1.3.3 id . . . . .	10
5.1.3.4 [] . . . . .	10
5.1.3.5 left . . . . .	10
5.1.3.6 num . . . . .	10
5.1.3.7 [struct] . . . . .	10
5.1.3.8 oper . . . . .	11
5.1.3.9 right . . . . .	11
5.1.3.10 stm . . . . .	11
5.1.3.11 [union] . . . . .	11
5.2 A_expList_ Struct Reference . . . . .	11
5.2.1 Detailed Description . . . . .	12
5.2.2 Member Enumeration Documentation . . . . .	12
5.2.2.1 anonymous enum . . . . .	12
5.2.3 Field Documentation . . . . .	12
5.2.3.1 head . . . . .	12
5.2.3.2 [] . . . . .	12
5.2.3.3 last . . . . .	12
5.2.3.4 [struct] . . . . .	12
5.2.3.5 tail . . . . .	12

5.2.3.6 [union] . . . . .	13
5.3 A_stm_ Struct Reference . . . . .	13
5.3.1 Detailed Description . . . . .	13
5.3.2 Member Enumeration Documentation . . . . .	13
5.3.2.1 anonymous enum . . . . .	13
5.3.3 Field Documentation . . . . .	14
5.3.3.1 [struct] . . . . .	14
5.3.3.2 [struct] . . . . .	14
5.3.3.3 exp . . . . .	14
5.3.3.4 exps . . . . .	14
5.3.3.5 id . . . . .	14
5.3.3.6 [] . . . . .	14
5.3.3.7 [struct] . . . . .	14
5.3.3.8 stm1 . . . . .	14
5.3.3.9 stm2 . . . . .	15
5.3.3.10 [union] . . . . .	15
5.4 int_and_table Struct Reference . . . . .	15
5.4.1 Field Documentation . . . . .	15
5.4.1.1 i . . . . .	15
5.4.1.2 tail . . . . .	15
5.5 table Struct Reference . . . . .	16
5.5.1 Field Documentation . . . . .	16
5.5.1.1 id . . . . .	16
5.5.1.2 tail . . . . .	16
5.5.1.3 value . . . . .	16
<b>6 File Documentation . . . . .</b>	<b>17</b>
6.1 src/SLP_interpreter/grammar_interpreter.cpp File Reference . . . . .	17
6.1.1 Function Documentation . . . . .	17
6.1.1.1 A_AssignStm() . . . . .	17
6.1.1.2 A_CompoundStm() . . . . .	17
6.1.1.3 A_EseqExp() . . . . .	17
6.1.1.4 A_IdExp() . . . . .	17
6.1.1.5 A_LastExpList() . . . . .	17
6.1.1.6 A_NumExp() . . . . .	18
6.1.1.7 A_OpExp() . . . . .	18
6.1.1.8 A_PairExpList() . . . . .	18
6.1.1.9 A_PrintStm() . . . . .	18
6.2 src/SLP_interpreter/grammar_interpreter.hpp File Reference . . . . .	18
6.2.1 Detailed Description . . . . .	18
6.2.2 Typedef Documentation . . . . .	18
6.2.2.1 A_exp . . . . .	18

6.2.2.2 A_expList	19
6.2.2.3 A_stm	19
6.2.3 Enumeration Type Documentation	19
6.2.3.1 A_binop	19
6.2.4 Function Documentation	19
6.2.4.1 A_AssignStm()	19
6.2.4.2 A_CompoundStm()	19
6.2.4.3 A_EseqExp()	19
6.2.4.4 A_IdExp()	20
6.2.4.5 A_LastExpList()	20
6.2.4.6 A_NumExp()	20
6.2.4.7 A_OpExp()	20
6.2.4.8 A_PairExpList()	20
6.2.4.9 A_PrintStm()	20
6.3 grammar_interpreter.hpp	20
6.4 src/SLP_interpreter/main.cpp File Reference	22
6.4.1 Function Documentation	22
6.4.1.1 interp()	22
6.4.1.2 interpExp()	22
6.4.1.3 interpExpList()	22
6.4.1.4 interpStm()	22
6.4.1.5 lookup()	22
6.4.1.6 main()	22
6.4.1.7 maxargs()	23
6.4.1.8 min()	23
6.4.1.9 prog_generator()	23
6.4.1.10 of SLP interpreter	23
6.4.1.11 prog_generator2()	23
6.4.1.12 update()	23
6.4.1.13 update_e()	23
6.4.1.14 update_el()	24
6.4.1.15 update_s()	24
6.5 src/SLP_interpreter/main.hpp File Reference	24
6.5.1 Typedef Documentation	24
6.5.1.1 IntAndTable_	24
6.5.1.2 Table_	24
6.5.2 Function Documentation	24
6.5.2.1 interp()	24
6.5.2.2 interpExp()	25
6.5.2.3 interpExpList()	25
6.5.2.4 interpStm()	25
6.5.2.5 maxargs()	25

---

6.5.2.6 Table()	25
6.5.2.7 update_e()	25
6.5.2.8 update_el()	26
6.5.2.9 update_s()	26
6.6 main.hpp	26
6.7 src/utls.cpp File Reference	27
6.7.1 Function Documentation	27
6.7.1.1 checked_malloc()	27
6.7.1.2 String()	27
6.8 src/utls.hpp File Reference	27
6.8.1 Typedef Documentation	27
6.8.1.1 string	27
6.8.2 Function Documentation	27
6.8.2.1 checked_malloc()	27
6.8.2.2 String()	28
6.9 utls.hpp	28
<b>Index</b>	<b>29</b>

# Chapter 1

## Topic Index

### 1.1 Topics

Here is a list of all topics with brief descriptions:

SLP_interpreter . . . . .	<a href="#">7</a>
Utils . . . . .	<a href="#">7</a>





## Chapter 2

# Struct Index

### 2.1 Struct List

Here are the structs with brief descriptions:

<a href="#">A_exp_</a> . . . . .	9
<a href="#">A_expList_</a> . . . . .	11
<a href="#">A_stm_</a> . . . . .	13
<a href="#">int_and_table</a> . . . . .	15
<a href="#">table</a> . . . . .	16



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">utils.cpp</a> . . . . .	27
src/ <a href="#">utils.hpp</a> . . . . .	27
src/SLP_interpreter/ <a href="#">grammar_interpreter.cpp</a> . . . . .	17
src/SLP_interpreter/ <a href="#">grammar_interpreter.hpp</a>	
Realize of grammar . . . . .	18
src/SLP_interpreter/ <a href="#">main.cpp</a> . . . . .	22
src/SLP_interpreter/ <a href="#">main.hpp</a> . . . . .	24



## Chapter 4

# Topic Documentation

### 4.1 SLP\_interpreter

an interpreter to Straight-Line Program(SLP) language

#### Files

- file [grammar\\_interpreter.hpp](#)  
*realize of grammar*

#### 4.1.1 Detailed Description

an interpreter to Straight-Line Program(SLP) language

#### 4.1.2 Function Documentation

##### 4.1.2.1 max()

```
template<typename T = int>
T max (
    T a,
    T b )
```

##### 4.1.3 `<tt>__main__</tt>` file

### 4.2 Utils

realize of [utils.hpp](#)

realize of [utils.hpp](#)

USE ASSERT!!!



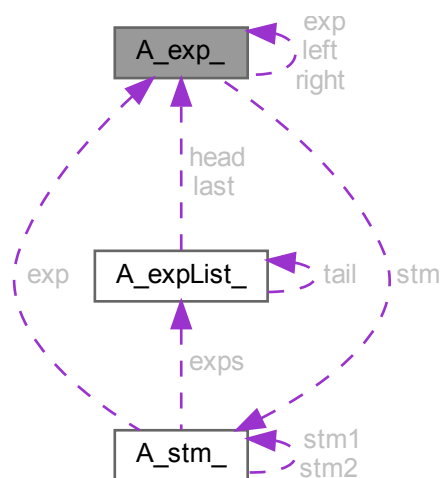
## Chapter 5

# Struct Documentation

### 5.1 A\_exp\_ Struct Reference

```
#include <grammar_interpreter.hpp>
```

Collaboration diagram for A\_exp\_:



#### Public Types

#### Data Fields

#### 5.1.1 Detailed Description

```
Exp -> id
Exp -> num
Exp -> Exp Binop Exp
Exp -> (Stm, Exp)
```

## 5.1.2 Member Enumeration Documentation

### 5.1.2.1 anonymous enum

anonymous enum

Enumerator

A_idExp	
A_numExp	
A_opExp	
A_eseqExp	

## 5.1.3 Field Documentation

### 5.1.3.1 [struct]

```
struct { ... } eseq
```

### 5.1.3.2 exp

A\_exp exp

### 5.1.3.3 id

string id

### 5.1.3.4 []

```
enum { ... } kind
```

### 5.1.3.5 left

A\_exp left

### 5.1.3.6 num

```
int num
```

### 5.1.3.7 [struct]

```
struct { ... } op
```



### 5.1.3.8 oper

`A_binop` oper

### 5.1.3.9 right

`A_exp` right

### 5.1.3.10 stm

`A_stm` stm

### 5.1.3.11 [union]

```
union { ... } u
```

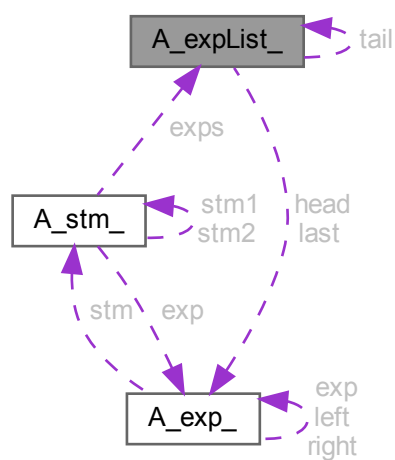
The documentation for this struct was generated from the following file:

- `src/SLP_interpreter/grammar_interpreter.hpp`

## 5.2 A\_expList\_ Struct Reference

```
#include <grammar_interpreter.hpp>
```

Collaboration diagram for A\_expList\_:



## Public Types

## Data Fields

### 5.2.1 Detailed Description

```
ExpList -> Exp, ExpList
ExpList -> Exp
```

### 5.2.2 Member Enumeration Documentation

#### 5.2.2.1 anonymous enum

anonymous enum

#### Enumerator

A_pairExpList	
A_lastExpList	

### 5.2.3 Field Documentation

#### 5.2.3.1 head

[A\\_exp](#) head

#### 5.2.3.2 []

```
enum { ... } kind
```

#### 5.2.3.3 last

[A\\_exp](#) last

#### 5.2.3.4 [struct]

```
struct { ... } pair
```

#### 5.2.3.5 tail

[A\\_expList](#) tail

## 5.2.3.6 [union]

```
union { ... } u
```

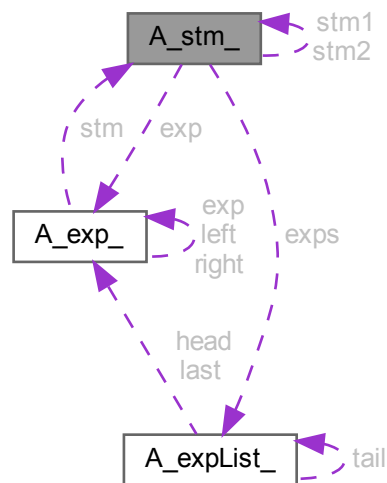
The documentation for this struct was generated from the following file:

- src/SLP\_interpreter/grammar\_interpreter.hpp

## 5.3 A\_stm\_ Struct Reference

```
#include <grammar_interpreter.hpp>
```

Collaboration diagram for A\_stm\_:



## Public Types

## Data Fields

## 5.3.1 Detailed Description

```
Stm -> Stm; Stm
Stm -> id := Stm
Stm -> print (ExpList)
```

## 5.3.2 Member Enumeration Documentation

## 5.3.2.1 anonymous enum

```
anonymous enum
```

**Enumerator**

A_compoundStm	
A_assignStm	
A_printStm	

**5.3.3 Field Documentation****5.3.3.1 [struct]**

```
struct { ... } assign
```

**5.3.3.2 [struct]**

```
struct { ... } compound
```

**5.3.3.3 exp**

```
A_exp exp
```

**5.3.3.4 exps**

```
A_expList exps
```

**5.3.3.5 id**

```
string id
```

**5.3.3.6 []**

```
enum { ... } kind
```

**5.3.3.7 [struct]**

```
struct { ... } print
```

**5.3.3.8 stm1**

```
A_stm stm1
```

### 5.3.3.9 stm2

```
A_stm stm2
```

### 5.3.3.10 [union]

```
union { ... } u
```

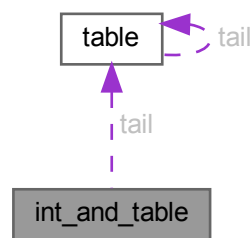
The documentation for this struct was generated from the following file:

- src/SLP\_interpreter/[grammar\\_interpreter.hpp](#)

## 5.4 int\_and\_table Struct Reference

```
#include <main.hpp>
```

Collaboration diagram for int\_and\_table:



### Data Fields

### 5.4.1 Field Documentation

#### 5.4.1.1 i

```
int i
```

#### 5.4.1.2 tail

```
Table_ tail
```

The documentation for this struct was generated from the following file:

- src/SLP\_interpreter/[main.hpp](#)

## 5.5 table Struct Reference

```
#include <main.hpp>
```

Collaboration diagram for table:



### Data Fields

#### 5.5.1 Field Documentation

##### 5.5.1.1 id

```
string id
```

##### 5.5.1.2 tail

```
Table_ tail
```

##### 5.5.1.3 value

```
int value
```

The documentation for this struct was generated from the following file:

- src/SLP\_interpreter/[main.hpp](#)

# Chapter 6

## File Documentation

### 6.1 src/SLP\_interpreter/grammar\_interpreter.cpp File Reference

#### 6.1.1 Function Documentation

##### 6.1.1.1 A\_AssignStm()

```
A_stm A_AssignStm (
    string id,
    A_exp exp )
```

##### 6.1.1.2 A\_CompoundStm()

```
A_stm A_CompoundStm (
    A_stm stm1,
    A_stm stm2 )
```

##### 6.1.1.3 A\_EseqExp()

```
A_exp A_EseqExp (
    A_stm stm,
    A_exp exp )
```

##### 6.1.1.4 A\_IdExp()

```
A_exp A_IdExp (
    string id )
```

##### 6.1.1.5 A\_LastExpList()

```
A_expList A_LastExpList (
    A_exp last )
```

### 6.1.1.6 A\_NumExp()

```
A_exp A_NumExp (
    int num )
```

### 6.1.1.7 A\_OpExp()

```
A_exp A_OpExp (
    A_exp left,
    A_binop oper,
    A_exp right )
```

### 6.1.1.8 A\_PairExpList()

```
A_expList A_PairExpList (
    A_exp head,
    A_expList tail )
```

### 6.1.1.9 A\_PrintStm()

```
A_stm A_PrintStm (
    A_expList exps )
```

## 6.2 src/SLP\_interpreter/grammar\_interpreter.hpp File Reference

realize of grammar

### 6.2.1 Detailed Description

realize of grammar

Grammar atoms: stm, exp, expList, id, num, ...

Grammar rules: A\_compoundStm, A\_assignStm, A\_printStm, ...

```
* For example, grammar atoms ga have rules gr1, gr2, ...
* using A_ga = ptr to struct A_ga_
* struct A_ga_:
*     field enum    -> rules tokens
*     field union   -> rules components
* constructo of rule:
* auto A_gr(A_ga components ...) -> pointer to struct A_ga;
*
```

### 6.2.2 Typedef Documentation

#### 6.2.2.1 A\_exp

```
using A_exp = A_exp_ *
```



### 6.2.2.2 A\_expList

```
using A_expList = A_expList_ *
```

### 6.2.2.3 A\_stm

```
using A_stm = A_stm_ *
```

grammars was defined to type  
for with data grammars, we use pointer to struct  
for without data grammars, we use enum

## 6.2.3 Enumeration Type Documentation

### 6.2.3.1 A\_binop

```
enum A_binop
```

Enumerator

A_plus	
A_minus	
A_times	
A_div	

## 6.2.4 Function Documentation

### 6.2.4.1 A\_AssignStm()

```
A_stm A_AssignStm (  
    string id,  
    A_exp exp )
```

### 6.2.4.2 A\_CompoundStm()

```
A_stm A_CompoundStm (  
    A_stm stm1,  
    A_stm stm2 )
```

### 6.2.4.3 A\_EseqExp()

```
A_exp A_EseqExp (  
    A_stm stm,  
    A_exp exp )
```

#### 6.2.4.4 A\_IdExp()

```
A_exp A_IdExp (
    string id )
```

#### 6.2.4.5 A\_LastExpList()

```
A_expList A_LastExpList (
    A_exp last )
```

#### 6.2.4.6 A\_NumExp()

```
A_exp A_NumExp (
    int num )
```

#### 6.2.4.7 A\_OpExp()

```
A_exp A_OpExp (
    A_exp left,
    A_binop oper,
    A_exp right )
```

#### 6.2.4.8 A\_PairExpList()

```
A_expList A_PairExpList (
    A_exp head,
    A_expList tail )
```

#### 6.2.4.9 A\_PrintStm()

```
A_stm A_PrintStm (
    A_expList exps )
```

### 6.3 grammar\_interpreter.hpp

[Go to the documentation of this file.](#)

```
00001
00026 #include "../utils.hpp"
00027
00028 using A_stm = struct A_stm_ *;
00029 using A_exp = struct A_exp_ *;
00030 using A_expList = struct A_expList_ *;
00031 enum A_binop
00032 {
00033     A_plus, // +
00034     A_minus, // -
00035     A_times, // *
00036     A_div // /
00037 };
00038
00048 struct A_stm_
00049 {
```

```

00050     enum
00051     {
00052         A_compoundStm, // Stm -> Stm; Stm
00053         A_assignStm,   // Stm -> id := Exp
00054         A_printStm     // Stm -> print(ExpList)
00055     } kind;
00056     union
00057     {
00058         struct
00059         {
00060             A_stm stm1, stm2;
00061         } compound;
00062         struct
00063         {
00064             string id;
00065             A_exp exp;
00066         } assign;
00067         struct
00068         {
00069             A_expList exps;
00070         } print;
00071     } u;
00072 };
00073
00074 A_stm A_CompoundStm(A_stm stm1, A_stm stm2);
00075 A_stm A_AssignStm(string id, A_exp exp);
00076 A_stm A_PrintStm(A_expList exps);
00077
00078 struct A_exp_
00079 {
00080     enum
00081     {
00082         A_idExp, // Exp -> id
00083         A_numExp, // Exp -> num
00084         A_opExp, // Exp -> Exp Binop Exp
00085         A_eseqExp // Exp -> (Stm, Exp)
00086     } kind;
00087     union
00088     {
00089         {
00090             string id;
00091             int num;
00092             struct
00093             {
00094                 A_exp left;
00095                 A_binop oper;
00096                 A_exp right;
00097             } op;
00098             struct
00099             {
00100                 A_stm stm;
00101                 A_exp exp;
00102             } eseq;
00103         } u;
00104     };
00105
00106     A_exp A_IdExp(string id);
00107     A_exp A_NumExp(int num);
00108     A_exp A_OpExp(A_exp left, A_binop oper, A_exp right);
00109     A_exp A_EseqExp(A_stm stm, A_exp exp);
00110
00111     struct A_expList_
00112     {
00113         enum
00114         {
00115             A_pairExpList, // ExpList -> Exp, ExpList
00116             A_lastExpList  // ExpList -> Exp
00117         } kind;
00118         union
00119         {
00120             struct
00121             {
00122                 A_exp head;
00123                 A_expList tail;
00124             } pair;
00125             A_exp last;
00126         } u;
00127     };
00128
00129     A_expList A_PairExpList(A_exp head, A_expList tail);
00130     A_expList A_LastExpList(A_exp last);

```

## 6.4 src/SLP\_interpreter/main.cpp File Reference

### 6.4.1 Function Documentation

#### 6.4.1.1 interp()

```
void interp (
    A_stm s )
```

the final SLP interpreter.

#### 6.4.1.2 interpExp()

```
IntAndTable_ interpExp (
    A_exp e,
    Table_ t )
```

interpret a exp e, and table t is the current state.

#### 6.4.1.3 interpExpList()

```
IntAndTable_ interpExpList (
    A_expList el,
    Table_ t )
```

interpret a explist el, and table t is the current state.

#### 6.4.1.4 interpStm()

```
Table_ interpStm (
    A_stm s,
    Table_ t )
```

interpret a stm s, and table t is the current state.

#### 6.4.1.5 lookup()

```
int lookup (
    Table_ t,
    string key )
```

find the identifier key's value in table t

#### 6.4.1.6 main()

```
int main ( )
```

#### 6.4.1.7 maxargs()

```
int maxargs (
    A_stm s )
```

Count print args number max.

#### 6.4.1.8 min()

```
template<typename T = int>
T min (
    T a,
    T b )
```

#### 6.4.1.9 prog\_generator()

```
A_stm prog_generator (
    void )
```

Example Tiger Code 1.

#### 6.4.1.10 of SLP interpreter

```
a := 5+3; b := (print(a, a-1), 10*a);
print(b); print(c1);
```

#### 6.4.1.11 prog\_generator2()

```
A_stm prog_generator2 (
    void )
```

Example Tiger Code 2.

```
a := 5+3; b := (print(a, a-1), 10*a); print(b);
print(c1, c2, c3, c4, c5, c6, c7, c8, c9, c10);
```

#### 6.4.1.12 update()

```
Table_ update (
    Table_ t1,
    string id2,
    int v2 )
```

update table t1 with new identifier id2 and value t2

#### 6.4.1.13 update\_e()

```
void update_e (
    A_exp e,
    int & count,
    int & history_max,
    bool & is_in_print,
    bool & is_out_print )
```

update count counter and history\_max record in Exp

#### 6.4.1.14 `update_el()`

```
void update_el (
    A_expList el,
    int & count,
    int & history_max,
    bool & is_in_print,
    bool & is_out_print )
```

update count counter and history\_max record in ExpList

#### 6.4.1.15 `update_s()`

```
void update_s (
    A_stm s,
    int & count,
    int & history_max,
    bool & is_in_print,
    bool & is_out_print )
```

update count counter and history\_max record in Stm

## 6.5 `src/SLP_interpreter/main.hpp` File Reference

### Data Structures

- struct [int\\_and\\_table](#)

### 6.5.1 Typedef Documentation

#### 6.5.1.1 `IntAndTable_`

```
using IntAndTable_ = int_and_table *
```

#### 6.5.1.2 `Table_`

```
using Table_ = table *
```

`table` and [int\\_and\\_table](#) are used to store identifiers.

A `table` is used to store a `stm` and it's value.

A [int\\_and\\_table](#) is combines a table and a value, which is the expression's return value.

In fact, these tables are linked note list.

### 6.5.2 Function Documentation

#### 6.5.2.1 `interp()`

```
void interp (
    A_stm s )
```

the final SLP interpreter.

### 6.5.2.2 interpExp()

```
IntAndTable_ interpExp (
    A_exp e,
    Table_ t )
```

interpret a exp e, and table t is the current state.

### 6.5.2.3 interpExpList()

```
IntAndTable_ interpExpList (
    A_exp e,
    Table_ t )
```

### 6.5.2.4 interpStm()

```
Table_ interpStm (
    A_stm s,
    Table_ t )
```

interpret a stm s, and table t is the current state.

### 6.5.2.5 maxargs()

```
int maxargs (
    A_stm s )
```

Count print args number max.

### 6.5.2.6 Table()

```
Table_ Table (
    string id,
    int value,
    struct table * tail )
```

### 6.5.2.7 update\_e()

```
void update_e (
    A_exp e,
    int & count,
    int & history_max,
    bool & is_in_print,
    bool & is_out_print )
```

update count counter and history\_max record in Exp

### 6.5.2.8 update\_el()

```
void update_el (
    A_expList el,
    int & count,
    int & history_max,
    bool & is_in_print,
    bool & is_out_print )
```

update count counter and history\_max record in ExpList

### 6.5.2.9 update\_s()

```
void update_s (
    A_stm s,
    int & count,
    int & history_max,
    bool & is_in_print,
    bool & is_out_print )
```

update count counter and history\_max record in Stm

## 6.6 main.hpp

[Go to the documentation of this file.](#)

```
00001
00006 #include "grammar_interpreter.hpp"
00007
00008 void update_e(A_exp e, int &count, int &history_max, bool &is_in_print, bool &is_out_print);
00009 void update_el(A_expList el, int &count, int &history_max, bool &is_in_print, bool &is_out_print);
00010 void update_s(A_stm s, int &count, int &history_max, bool &is_in_print, bool &is_out_print);
00011 int maxargs(A_stm);
00012
00013 void interp(A_stm);
00014
00024 using Table_ = struct table *;
00025 using IntAndTable_ = struct int_and_table *;
00026
00027 struct table {
00028     string id;
00029     int value;
00030     Table_ tail;
00031 };
00032
00033 Table_ Table(string id, int value, struct table *tail) {
00034     Table_ t = (Table_)checked_malloc(sizeof(*t));
00035     *t = {.id = id, .value = value, .tail = tail};
00036     return t;
00037 }
00038
00039 struct int_and_table {
00040     int i;
00041     Table_ tail;
00042 };
00043
00044 Table_ interpStm(A_stm s, Table_ t);
00045 IntAndTable_ interpExp(A_exp e, Table_ t);
00046 IntAndTable_ interpExpList(A_exp e, Table_ t);
00047 void interp(A_stm);
```



## 6.7 src/utils.cpp File Reference

### 6.7.1 Function Documentation

#### 6.7.1.1 checked\_malloc()

```
void* checked_malloc (
    std::size_t size )
```

Don't use new/delete(C++), malloc/calloc/free(C)

In C++, memory allocator in <cstdlib>

#### Warning

malloc will return NULL (C) or nullptr (C++>=11)

#### 6.7.1.2 String()

```
string String (
    string origin )
```

string constructor

## 6.8 src/utils.hpp File Reference

### 6.8.1 Typedef Documentation

#### 6.8.1.1 string

```
using string = const char *
```

#### Warning

we will store it in heap

### 6.8.2 Function Documentation

#### 6.8.2.1 checked\_malloc()

```
void* checked_malloc (
    std::size_t size )
```

Don't use new/delete(C++), malloc/calloc/free(C)

noticed that, we will allocate memory for struct with union member because of the uncertain size of union, the default constructor is deleted, so new cannot used to create an struct object which with union member.

for some reason, we do not use new/delete in all project code, and we will use checked\_malloc forever instead of malloc/calloc, and we never use free because of garbage collector. (Reason can be found in textbook.)

**Parameters**

<code>std::size_t</code>	the size will allocated
--------------------------	-------------------------

**Returns**

a `void *` pointer point to memory which is allocated by this call.

In C++, memory allocator in `<cstdlib>`

**Warning**

`malloc` will return `NULL` (C) or `nullptr` (C++>=11)

**6.8.2.2 String()**

```
string String (
    string origin )
```

string constructor

**6.9 utils.hpp**

[Go to the documentation of this file.](#)

```
00001
00002 #include <assert.h>
00003 #include <cstdlib>
00004 #include <string>
00005
00007 using string = const char *;
00009 string String(string);
00010
00024 void *checked_malloc(std::size_t);
```

# Index

- A\_AssignStm
  - grammar\_interpreter.cpp, [17](#)
  - grammar\_interpreter.hpp, [19](#)
- A\_assignStm
  - A\_stm\_, [14](#)
- A\_binop
  - grammar\_interpreter.hpp, [19](#)
- A\_CompoundStm
  - grammar\_interpreter.cpp, [17](#)
  - grammar\_interpreter.hpp, [19](#)
- A\_compoundStm
  - A\_stm\_, [14](#)
- A\_div
  - grammar\_interpreter.hpp, [19](#)
- A\_EseqExp
  - grammar\_interpreter.cpp, [17](#)
  - grammar\_interpreter.hpp, [19](#)
- A\_eseqExp
  - A\_exp\_, [10](#)
- A\_exp
  - grammar\_interpreter.hpp, [18](#)
- A\_exp\_, [9](#)
  - A\_eseqExp, [10](#)
  - A\_idExp, [10](#)
  - A\_numExp, [10](#)
  - A\_opExp, [10](#)
  - eseq, [10](#)
  - exp, [10](#)
  - id, [10](#)
  - kind, [10](#)
  - left, [10](#)
  - num, [10](#)
  - op, [10](#)
  - oper, [10](#)
  - right, [11](#)
  - stm, [11](#)
  - u, [11](#)
- A\_expList
  - grammar\_interpreter.hpp, [18](#)
- A\_expList\_, [11](#)
  - A\_lastExpList, [12](#)
  - A\_pairExpList, [12](#)
  - head, [12](#)
  - kind, [12](#)
  - last, [12](#)
  - pair, [12](#)
  - tail, [12](#)
  - u, [12](#)
- A\_IdExp
  - grammar\_interpreter.cpp, [17](#)
  - grammar\_interpreter.hpp, [19](#)
- A\_idExp
  - A\_exp\_, [10](#)
- A\_LastExpList
  - grammar\_interpreter.cpp, [17](#)
  - grammar\_interpreter.hpp, [20](#)
- A\_lastExpList
  - A\_expList\_, [12](#)
- A\_minus
  - grammar\_interpreter.hpp, [19](#)
- A\_NumExp
  - grammar\_interpreter.cpp, [17](#)
  - grammar\_interpreter.hpp, [20](#)
- A\_numExp
  - A\_exp\_, [10](#)
- A\_OpExp
  - grammar\_interpreter.cpp, [18](#)
  - grammar\_interpreter.hpp, [20](#)
- A\_opExp
  - A\_exp\_, [10](#)
- A\_PairExpList
  - grammar\_interpreter.cpp, [18](#)
  - grammar\_interpreter.hpp, [20](#)
- A\_pairExpList
  - A\_expList\_, [12](#)
- A\_plus
  - grammar\_interpreter.hpp, [19](#)
- A\_PrintStm
  - grammar\_interpreter.cpp, [18](#)
  - grammar\_interpreter.hpp, [20](#)
- A\_printStm
  - A\_stm\_, [14](#)
- A\_stm
  - grammar\_interpreter.hpp, [19](#)
- A\_stm\_, [13](#)
  - A\_assignStm, [14](#)
  - A\_compoundStm, [14](#)
  - A\_printStm, [14](#)
  - assign, [14](#)
  - compound, [14](#)
  - exp, [14](#)
  - exps, [14](#)
  - id, [14](#)
  - kind, [14](#)
  - print, [14](#)
  - stm1, [14](#)
  - stm2, [14](#)
  - u, [15](#)

- A\_times
  - grammar\_interpreter.hpp, 19
- assign
  - A\_stm\_, 14
- checked\_malloc
  - utils.cpp, 27
  - utils.hpp, 27
- compound
  - A\_stm\_, 14
- eseq
  - A\_exp\_, 10
- exp
  - A\_exp\_, 10
  - A\_stm\_, 14
- exps
  - A\_stm\_, 14
- grammar\_interpreter.cpp
  - A\_AssignStm, 17
  - A\_CompoundStm, 17
  - A\_EseqExp, 17
  - A\_IdExp, 17
  - A\_LastExpList, 17
  - A\_NumExp, 17
  - A\_OpExp, 18
  - A\_PairExpList, 18
  - A\_PrintStm, 18
- grammar\_interpreter.hpp
  - A\_AssignStm, 19
  - A\_binop, 19
  - A\_CompoundStm, 19
  - A\_div, 19
  - A\_EseqExp, 19
  - A\_exp, 18
  - A\_expList, 18
  - A\_IdExp, 19
  - A\_LastExpList, 20
  - A\_minus, 19
  - A\_NumExp, 20
  - A\_OpExp, 20
  - A\_PairExpList, 20
  - A\_plus, 19
  - A\_PrintStm, 20
  - A\_stm, 19
  - A\_times, 19
- head
  - A\_expList\_, 12
- i
  - int\_and\_table, 15
- id
  - A\_exp\_, 10
  - A\_stm\_, 14
  - table, 16
- int\_and\_table, 15
  - i, 15
  - tail, 15
- IntAndTable\_
  - main.hpp, 24
- interp
  - main.cpp, 22
  - main.hpp, 24
- interpExp
  - main.cpp, 22
  - main.hpp, 24
- interpExpList
  - main.cpp, 22
  - main.hpp, 25
- interpStm
  - main.cpp, 22
  - main.hpp, 25
- kind
  - A\_exp\_, 10
  - A\_expList\_, 12
  - A\_stm\_, 14
- last
  - A\_expList\_, 12
- left
  - A\_exp\_, 10
- lookup
  - main.cpp, 22
- main
  - main.cpp, 22
- main.cpp
  - interp, 22
  - interpExp, 22
  - interpExpList, 22
  - interpStm, 22
  - lookup, 22
  - main, 22
  - maxargs, 22
  - min, 23
  - prog\_generator, 23
  - prog\_generator2, 23
  - update, 23
  - update\_e, 23
  - update\_el, 23
  - update\_s, 24
- main.hpp
  - IntAndTable\_, 24
  - interp, 24
  - interpExp, 24
  - interpExpList, 25
  - interpStm, 25
  - maxargs, 25
  - Table, 25
  - Table\_, 24
  - update\_e, 25
  - update\_el, 25
  - update\_s, 26
- max
  - SLP\_interpreter, 7

maxargs  
    main.cpp, [22](#)  
    main.hpp, [25](#)  
min  
    main.cpp, [23](#)  
num  
    A\_exp\_, [10](#)  
op  
    A\_exp\_, [10](#)  
oper  
    A\_exp\_, [10](#)  
pair  
    A\_expList\_, [12](#)  
print  
    A\_stm\_, [14](#)  
prog\_generator  
    main.cpp, [23](#)  
prog\_generator2  
    main.cpp, [23](#)  
right  
    A\_exp\_, [11](#)  
SLP\_interpreter, [7](#)  
    max, [7](#)  
src/SLP\_interpreter/grammar\_interpreter.cpp, [17](#)  
src/SLP\_interpreter/grammar\_interpreter.hpp, [18](#), [20](#)  
src/SLP\_interpreter/main.cpp, [22](#)  
src/SLP\_interpreter/main.hpp, [24](#), [26](#)  
src/Utils.cpp, [27](#)  
src/Utils.hpp, [27](#), [28](#)  
stm  
    A\_exp\_, [11](#)  
stm1  
    A\_stm\_, [14](#)  
stm2  
    A\_stm\_, [14](#)  
String  
    utils.cpp, [27](#)  
    utils.hpp, [28](#)  
string  
    utils.hpp, [27](#)  
Table  
    main.hpp, [25](#)  
table, [16](#)  
    id, [16](#)  
    tail, [16](#)  
    value, [16](#)  
Table\_  
    main.hpp, [24](#)  
tail  
    A\_expList\_, [12](#)  
    int\_and\_table, [15](#)  
    table, [16](#)  
u  
    A\_exp\_, [11](#)  
    A\_expList\_, [12](#)  
    A\_stm\_, [15](#)  
update  
    main.cpp, [23](#)  
update\_e  
    main.cpp, [23](#)  
    main.hpp, [25](#)  
update\_el  
    main.cpp, [23](#)  
    main.hpp, [25](#)  
update\_s  
    main.cpp, [24](#)  
    main.hpp, [26](#)  
Utils, [7](#)  
utils.cpp  
    checked\_malloc, [27](#)  
    String, [27](#)  
utils.hpp  
    checked\_malloc, [27](#)  
    String, [28](#)  
    string, [27](#)  
value  
    table, [16](#)