Prog2 NHF - Spreadsheet

Készítette Doxygen 1.9.1

1. fejezet

Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

2. fejezet

Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. fejezet

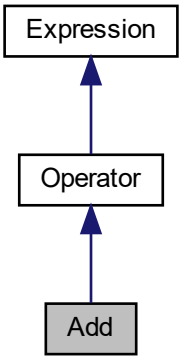
Osztályok dokumentációja

3.1. Add osztályreferencia

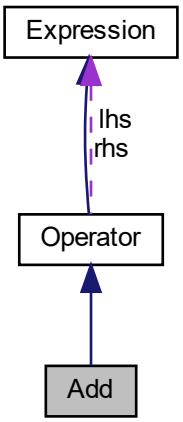
Összeadás műveletet reprezentáló osztály.

`#include <expression.hpp>`

Az Add osztály származási diagramja:



Az Add osztály együttműködési diagramja:



Publikus tagfüggvények

- **Add** (Expression ∗lhs, Expression ∗rhs)
- double eval () const

  rekurzívan kiértékeli a kifejezést.
- std::string show () const

  kifejezés megjelenítése std::string-ként
- Expression ∗ copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.1.1. Részletes leírás

Összeadás műveletet reprezentáló osztály.

3.1.2. Tagfüggvények dokumentációja

3.1.2.1. eval()

```
double Add::eval ( ) const  [inline], [virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

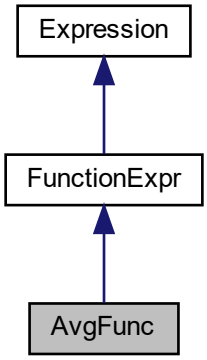
- srcs/expression.hpp

3.2. AvgFunc osztályreferencia

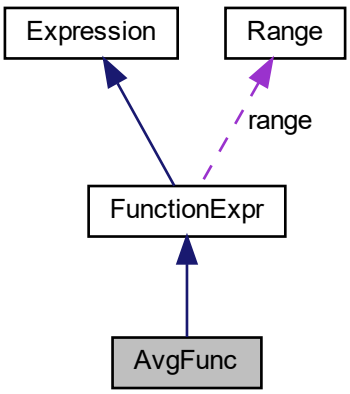
Tartomány átlagát vevő függvény osztály.

`#include <expression.hpp>`

Az AvgFunc osztály származási diagramja:



Az AvgFunc osztály együttműködési diagramja:



Publikus tagfüggvények

- **AvgFunc** (Range r)
- **AvgFunc** (CellRefExpr ∗topCell, CellRefExpr ∗bottomCell)
- double eval () const

rekurzívan kiértékeli a kifejezést.

- std::string show () const

  kifejezés megjelenítése std::string-ként

- Expression * copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.2.1. Részletes leírás

Tartomány átlagát vevő függvény osztály.

3.2.2. Tagfüggvények dokumentációja

3.2.2.1. eval()

```
double AvgFunc::eval ( ) const  [virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/expression.hpp
- srcs/expression.cpp

3.3. CellId osztályreferencia

Cellát azonosító sor- és oszlopadat eltárolására szolgáló osztály.

```
#include <expression.hpp>
```

Publikus tagfüggvények

- CellId (std::string col, int row)

  konstruktor oszlopjelölő betű és sorszám megadásával

- CellId (std::string)

  konstruktor "[oszlopbetű][sorszám]" formátumú bemenettel

- int getColNum () const

  oszlopszám lekérdezése

- int getRow () const

  sorszám lekérdezése

- void setColNum (int c)

  oszlopszám beállítása

- void setRow (int r)

  sorszám beállítása

- std::string colLetter () const

  oszlopbetű lekérdezése

3.3.1. Részletes leírás

Cellát azonosító sor- és oszlopadat eltárolására szolgáló osztály.

oszlop és sorindexeket is 1-től indexelve tárolja, paraméterként is így várja.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

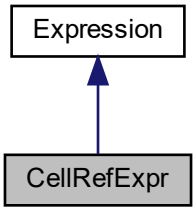
- srcs/expression.hpp
- srcs/expression.cpp

3.4. CellRefExpr osztályreferencia

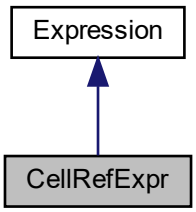
Cellahivatkozást reprezentáló kifejezés osztály.

`#include <expression.hpp>`

A CellRefExpr osztály származási diagramja:



A CellRefExpr osztály együttműködési diagramja:



Publikus tagfüggvények

- CellRefExpr (std::string col, int row, Sheet ∗sh=NULL, bool absCol=false, bool absRow=false)
  konstruktor oszlopjelölő betű és sorszám megadásával
- CellRefExpr (std::string str, Sheet ∗sh=NULL, bool absCol=false, bool absRow=false)
  konstruktor "[oszlopbetű][sorszám]" formátumú bemenettel
- std::string getCol () const
  oszlopbetű lekérdezése
- int getRow () const
  sorszám lekérdezése
- Sheet ∗ getSheet () const
  hivatkozás által mutatott tábla lekérdezése
- ExprPointer ∗ getPtr () const
  hivatkozás által mutatott cellára mutató pointer lekérdezése
- bool getAbsCol () const
  oszlop abszolút voltának lekérdezése
- bool getAbsRow () const
  sor abszolút voltának lekérdezése
- double eval () const
  hivatkozás által mutatott cella kiértékelése
- void checkCyclic (std::vector< Expression ∗ >) const
  ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát
- std::string show () const
  kifejezés megjelenítése std::string-ként
- CellRefExpr ∗ copy () const
  dinamikusan foglalt memóriaterületen visszaadott másolat
- void shift (int dx, int dy)
  Eltolja a hivatkozást adott sorral és oszloppal, amennyiben a sor/oszlop nem abszolút.
- void relocate (Sheet ∗shp)
  a cellahivatkozás célpontját áthelyezi egy másik számolótáblára

3.4.1. Részletes leírás

Cellahivatkozást reprezentáló kifejezés osztály.

A hivatkozás egy tábla (Sheet) egy cellájára mutathat oszlop és sor megadásával. Mind az oszlopa, mind a sora egymástól független lehetnek abszolútak.

3.4.2. Konstruktorok és destruktorok dokumentációja

3.4.2.1. CellRefExpr() [1/2]

```
CellRefExpr::CellRefExpr (
            std::string col,
            int row,
            Sheet * sh = NULL,
            bool absCol = false,
            bool absRow = false )  [inline]
```

konstruktor oszlopjelölő betű és sorszám megadásával

Paraméterek

col	- oszlopbetű
row	- sorszám
sh	- tábla, amelyre a hivatkozás mutat
absCol	- abszolút hivatkozás-e az oszlop
absRow	- abszolút hivatkozás-e a sor

3.4.2.2. CellRefExpr() [2/2]

```
CellRefExpr::CellRefExpr (
            std::string str,
            Sheet * sh = NULL,
            bool absCol = false,
            bool absRow = false )  [inline]
```

konstruktor "[oszlopbetű][sorszám]" formátumú bemenettel

Paraméterek

str	- cella jelölője ("[oszlopbetű][sorszám]")
sh	- tábla, amelyre a hivatkozás mutat
absCol	- abszolút hivatkozás-e az oszlop
absRow	- abszolút hivatkozás-e a sor

3.4.3. Tagfüggvények dokumentációja

3.4.3.1. checkCyclic()

```
void CellRefExpr::checkCyclic (
            std::vector< Expression * > ) const  [virtual]
```

ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát

amennyiben igen, eval_error kivételt dob

Paraméterek

ps	- a kifejezésben korábban hivatkozott cellák, amire ismét hivatkozva körkörös hivatkozást kapunk

Megvalósítja a következőket: Expression.

3.4.3.2. shift()

```
void CellRefExpr::shift (
            int dx,
            int dy )  [virtual]
```

Eltolja a hivatkozást adott sorral és oszloppal, amennyiben a sor/oszlop nem abszolút.

Paraméterek

dx	- sor eltolásának mértéke (akár negatív)
dy	- oszlop eltolásának mértéke (akár negatív)

Újraimplementált ősök: Expression.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/expression.hpp
- srcs/expression.cpp

3.5. Console osztályreferencia

Felhasználói felület biztosítására szolgáló osztály.

```
#include <console.hpp>
```

Publikus tagfüggvények

- Console ()
  alapértelmezett konstruktor, input és outputstream-je a std::cin és std::cout
- Console (Sheet sh, std::ostream &os, std::istream &is)
  konstruktor tábla, input- és outputstreamek megadásával
- Console (std::ostream &os, std::istream &is)
  konstruktor csak input- és outputstreamek megadásával
- bool isClosed () const
  visszaadja, bezárták-e a konzolt
- void help ()
  kiírja az os-re az elérhető parancsokat
- void createNew ()
  új táblát hoz létre (ha volt előző, azt eldobja)
- void resize ()
  átméretezi a táblát, ha kisebb lesz, a fennmaradó adat elveszik.
- void print ()
  kiírja az os-re a tábla tartalmát oszlop- és sorszámokkal
- void exportValues ()
  is-ről bekért fájlnevű fájlba kiírja a táblában tárolt értékeket vesszővel elválasztva
- void save ()
  is-ről bekért fájlnevű fájlba kiírja a táblában tárolt kifejezéseket vesszővel elválasztva

- void load ()

  is-ről bekért fájlnevű fájlból beolvassa a vesszővel elválasztott kifejezéseket
- void set ()
- void pull ()

  automatikusan kitölti a kezdőcellában található értékkel a cellákat a második paraméterben kapott celláig egy téglalapban
- void show ()

  kiírja az os-re a is-ről olvasott cella tartalmát és értékét
- void exit ()

  bezárja a konzolt
- void readCommand ()

  beolvassa és értelmezi az is-re beírt parancs nevét, és meghívja a megfelelő tagfüggvényt

3.5.1. Részletes leírás

Felhasználói felület biztosítására szolgáló osztály.

A adott input- és outputstream-ről, a parancsokat tud beolvasni, illetve a parancsok eredményét ki tudja írni. Az osztály mindig egy darab táblázatot tartalmaz, a kapott utasításokat ezen hajtja végre. A parancsok kulcsszavait a readCommand tagfüggvény értelmezi, és ő hívja meg a megfelelő parancsot lekezelő tagfüggvényt. Az adott tagfüggvény az inputstreamről beolvassa a parancs paramétereit és végrehajtja a azt. Ha a felhasználó szintaktikailag hibás parancsot ad, akkor a konzol kapja el a program által generált kivételeket, és hibaüzenetet ír az outputstreamre.

3.5.2. Tagfüggvények dokumentációja

3.5.2.1. createNew()

```
void Console::createNew ( )
```

új táblát hoz létre (ha volt előző, azt eldobja)

paramétereit az is-ről olvassa: új tábla szélesség és magassága

3.5.2.2. pull()

```
void Console::pull ( )
```

automatikusan kitölti a kezdőcellában található értékkel a cellákat a második paraméterben kapott celláig egy téglalapban

a kezdőcellában található kifejezést átmásolja a két cella által meghatározott téglalap minden cellájába, ezen felül minden nem abszolút hivatkozást eltol a kezdőcellától vett relatív pozíciójának megfelelően (ld. CellRefExpr::shift)

3.5.2.3. readCommand()

```
void Console::readCommand ( )
```

beolvassa és értelmezi az is-re beírt parancs nevét, és meghívja a megfelelő tagfüggvényt

ha helytelen parancsnevet kap, hibaüzenetet ír az os-re

3.5.2.4. resize()

```
void Console::resize ( )
```

átméretezi a táblát, ha kisebb lesz, a fennmaradó adat elveszik.

paramétereit az is-ről olvassa: tábla új szélesség és magassága

3.5.2.5. set()

```
void Console::set ( )
```

is-ről bekért cellába beállítja a megadott kifejezést (amennyiben szintaktikailag helyes)

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

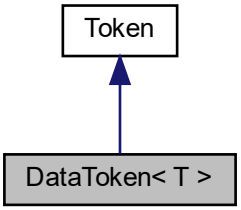
- srcs/console.hpp
- srcs/console.cpp

3.6. DataToken< T > osztálysablon-referencia

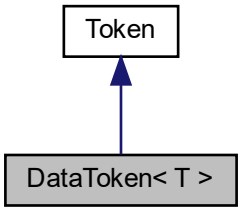
Adattartalommal rendelkező tokenek osztálya.

```
#include <token.hpp>
```

A DataToken< T > osztály származási diagramja:



A DataToken< T > osztály együttműködési diagramja:



Publikus tagfüggvények

- DataToken (Token_type tt, T s)

  konstruktor típus és adat megadásával
- T getContent () const

  belső adat lekérdezése
- Token ∗ copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.6.1. Részletes leírás

template<typename T>
class DataToken< T >

Adattartalommal rendelkező tokenek osztálya.

A token osztály leszármazottja az olyan tokeneknek, amelyek a típusokon felül más adattartalommal is rendelkeznek.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

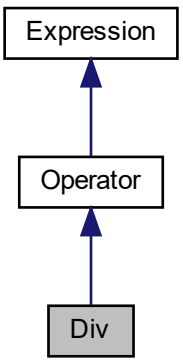
- srcs/token.hpp

3.7. Div osztályreferencia

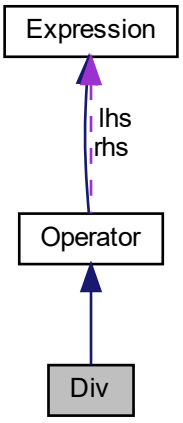
Osztás műveletet reprezentáló osztály.

`#include <expression.hpp>`

A Div osztály származási diagramja:



A Div osztály együttműködési diagramja:



Publikus tagfüggvények

- **Div** (Expression ∗lhs, Expression ∗rhs)
- double eval () const

rekurzívan kiértékeli a kifejezést.

- std::string show () const

  kifejezés megjelenítése std::string-ként

- Expression ∗ copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.7.1. Részletes leírás

Osztás műveletet reprezentáló osztály.

3.7.2. Tagfüggvények dokumentációja

3.7.2.1. eval()

```
double Div::eval ( ) const  [inline], [virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

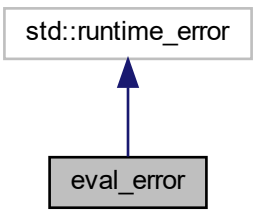
- srcs/expression.hpp

3.8. eval_error osztályreferencia

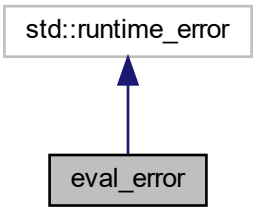
Saját exception osztály a kifejezések kiértékelésekor felmerülő hibákra.

```
#include <exceptions.hpp>
```

Az eval_error osztály származási diagramja:



Az eval_error osztály együttműködési diagramja:



3.8.1. Részletes leírás

Saját exception osztály a kifejezések kiértékelésekor felmerülő hibákra.

A szintakikai hiba részleteit a .what() tagfüggvényel kaphatjuk meg.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

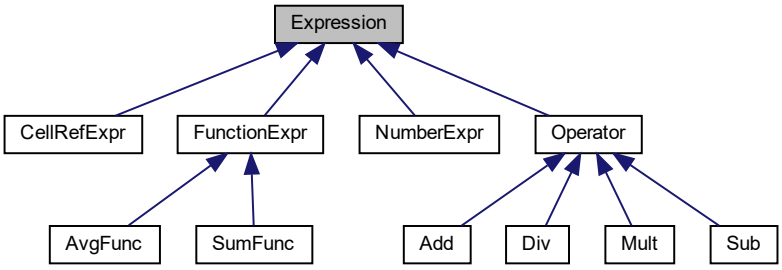
- srcs/exceptions.hpp

3.9. Expression osztályreferencia

Kifejezések absztrakt alaposztálya.

`#include <expression_core.hpp>`

Az Expression osztály származási diagramja:



Publikus tagfüggvények

- virtual double eval () const =0

  rekurzívan kiértékeli a kifejezést.
- virtual void checkCyclic (std::vector< Expression * >) const =0

  ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát
- virtual double safeEval (std::vector< Expression * > ps) const

  rekurzívan kiértékeli a kifejezést, körkörös hivatkozásra is eval_error hibát dob
- virtual std::string show () const =0

  kifejezés megjelenítése std::string-ként
- virtual Expression * copy () const =0

  dinamikusan foglalt memóriaterületen visszaadott másolat
- virtual void shift (int, int)

  rekurzívan minden hivatkozást adott oszlop- és sorszámmal eltol
- virtual void relocate (Sheet *)

  a kifejezésben található hivatkozások célpontját áthelyezi egy másik számolótáblára

3.9.1. Részletes leírás

Kifejezések absztrakt alaposztálya.

3.9.2. Tagfüggvények dokumentációja

3.9.2.1. checkCyclic()

```
virtual void Expression::checkCyclic (
            std::vector< Expression * >  ) const  [pure virtual]
```

ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát

amennyiben igen, eval_error kivételt dob

Paraméterek

ps	- a kifejezésben korábban hivatkozott cellák, amire ismét hivatkozva körkörös hivatkozást kapunk

Megvalósítják a következők: NumberExpr, FunctionExpr, CellRefExpr és Operator.

3.9.2.2. eval()

```
virtual double Expression::eval ( ) const  [pure virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítják a következők: NumberExpr, Sub, Add, Div, Mult, SumFunc, AvgFunc és CellRefExpr.

3.9.2.3. safeEval()

```
virtual double Expression::safeEval (
            std::vector< Expression * > ps ) const  [inline], [virtual]
```

rekurzívan kiértékeli a kifejezést, körkörös hivatkozásra is eval_error hibát dob

Paraméterek

ps	- a kifejezésben korábban hivatkozott cellák, amire ismét hivatkozva körkörös hivatkozást kapunk

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- srcs/expression_core.hpp

3.10. ExprPointer osztályreferencia

A kifejezésekre mutató pointerek wrapper osztálya.

```
#include <expression_core.hpp>
```

Publikus tagfüggvények

- ExprPointer (Expression *p=NULL)
  konstruktor pointer inicializálásával
- ExprPointer (const ExprPointer &rhs)
  másoló konstruktor
- operator Expression * () const
  castolás Expression-ra*
- ExprPointer & operator= (const ExprPointer &rhs)
  értékadás a másik kifejezés rekurzív másolásával.
- bool operator== (const ExprPointer &rhs) const
  egyenlőség másik ExprPointer-el
- bool operator== (Expression *p)
  egyenlőség Expression-al*
- Expression * operator-> () const
  becsomagolt pointer adatainak és függvényeinek elérése nyíllal
- virtual double evalMe ()
  kiértékeli az adott kifejezést úgy, hogy, a körkörös hivatkozások keresése tőle indul
- ~ExprPointer ()
  felszabadítja a pointert

3.10.1. Részletes leírás

A kifejezésekre mutató pointerek wrapper osztálya.

Az Expression absztrakt osztályból származtatott osztályok példányait heterogén kollekciókban pointerként tudjuk tárolni. Azonban ilyenkor ügyelni kell a dinamikusan foglalt memóriaterületekre, a pointerek másolására, értékadásra. Az ExprPointer osztály ezt hivatott lekezelni, azáltal, hogy a másolja és felszabadítja a pointereket, de az eredeti funkciójukat is megtartja. Így minden kifejezést gyakorlatilag sima osztálypéldányként tudunk kezelni (pointer helyett).

Ez a dokumentáció az osztályról a következő fájl alapján készült:

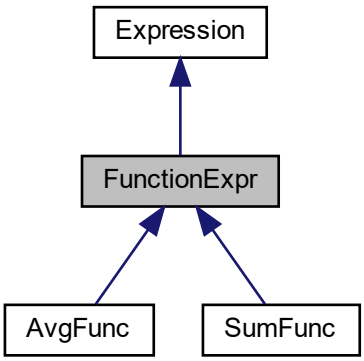
- srcs/expression_core.hpp

3.11. FunctionExpr osztályreferencia

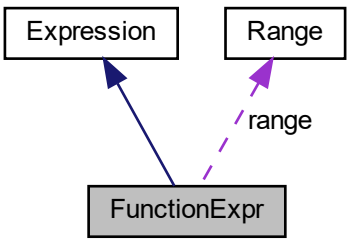
Tartományon elvégezhető függvénykifejezések absztrakt osztálya.

`#include <expression.hpp>`

A FunctionExpr osztály származási diagramja:



A FunctionExpr osztály együttműködési diagramja:



Publikus tagfüggvények

- FunctionExpr (Range r)
  konstruktor
- FunctionExpr (CellRefExpr *topCell, CellRefExpr *bottomCell)
  konstruktor
- void checkCyclic (std::vector< Expression * >) const
  ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát
- void shift (int dx, int dy)
  rekurzívan minden hivatkozást adott oszlop- és sorszámmal eltol
- void relocate (Sheet *shp)
  a kifejezésben található hivatkozások célpontját áthelyezi egy másik számolótáblára

Statikus publikus tagfüggvények

- static FunctionName parseFname (std::string name)
  értelmezi a függvények neveit (case sensitive)
- static FunctionExpr * newFunctionExpr (FunctionName fn, CellRefExpr *topCell, CellRefExpr *bottomCell)
  létrehoz egy megfelelő típusú függvényt a neve alapján

Védett attribútumok

- Range range
  tartomány, melyen a függvény végrehajtódik

3.11.1. Részletes leírás

Tartományon elvégezhető függvénykifejezések absztrakt osztálya.

3.11.2. Tagfüggvények dokumentációja

3.11.2.1. checkCyclic()

```
void FunctionExpr::checkCyclic (
            std::vector< Expression * >  ) const  [virtual]
```

ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát

amennyiben igen, eval_error kivételt dob

Paraméterek

ps	- a kifejezésben korábban hivatkozott cellák, amire ismét hivatkozva körkörös hivatkozást kapunk

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/expression.hpp
- srcs/expression.cpp

3.12. Range::iterator osztályreferencia

Tartományt sorfolytonosan bejáró iterátor.

`#include <expression.hpp>`

Publikus tagfüggvények

- iterator ()
- iterator (size_t rw, size_t tw, ExprPointer *bp)
  konstruktor
- ExprPointer & operator* () const
  iterátor tartalmának kiolvasása, runtime_error-t dob ha üres iterátorból olvasunk
- ExprPointer * operator-> () const
  iterátor tartalmának tagjainak elérése
- bool operator== (const ExprPointer *ep) const
  egyenlőség ExprPointer-el*
- bool operator== (const iterator &it) const
  egyenlőség egy másik iterátorral
- bool operator!= (const iterator &it) const
  egyenlőtlenség egy másik iterátorral
- iterator & operator++ ()
  preinkremens
- iterator operator++ (int)
  posztinkremens

3.12.1. Részletes leírás

Tartományt sorfolytonosan bejáró iterátor.

3.12.2. Konstruktorok és destruktorok dokumentációja

3.12.2.1. iterator() [1/2]

`Range::iterator::iterator ( )  [inline]`

üres iterátor létrehozása

3.12.2.2. iterator() [2/2]

```
Range::iterator::iterator (
            size_t rw,
            size_t tw,
            ExprPointer * bp )  [inline]
```

konstruktor

csak a tartománybeli sorok elejéről lehet indítani az iterátort

Paraméterek

rw	- a bejárandó tartomány szélessége
tw	- a bejárandó tartományt taralmazó tábla szélessége
bp	- a cella, melyre az iterátor kezdetben mutat

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

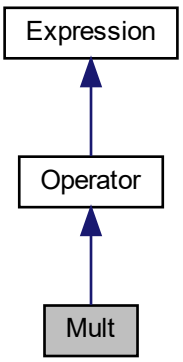
- srcs/expression.hpp
- srcs/expression.cpp

3.13. Mult osztályreferencia

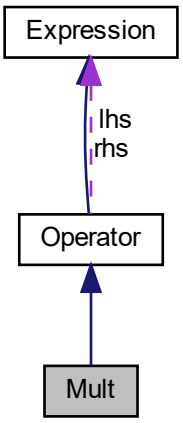
Szorzás műveletet reprezentáló osztály.

`#include <expression.hpp>`

A Mult osztály származási diagramja:



A Mult osztály együttműködési diagramja:



Publikus tagfüggvények

- **Mult** (Expression ∗lhs, Expression ∗rhs)
- double eval () const

  rekurzívan kiértékeli a kifejezést.
- std::string show () const

  kifejezés megjelenítése std::string-ként
- Expression ∗ copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.13.1. Részletes leírás

Szorzás műveletet reprezentáló osztály.

3.13.2. Tagfüggvények dokumentációja

3.13.2.1. eval()

```
double Mult::eval ( ) const  [inline], [virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

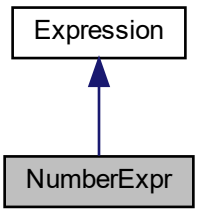
- srcs/expression.hpp

3.14. NumberExpr osztályreferencia

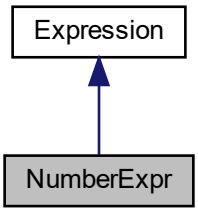
Valós számokat tároló kifejezés osztály.

`#include <expression_core.hpp>`

A NumberExpr osztály származási diagramja:



A NumberExpr osztály együttműködési diagramja:



Publikus tagfüggvények

- NumberExpr (double v)
  konstruktor
- double eval () const
  kifejezés kiértékelése - érték visszaadása
- void checkCyclic (std::vector< Expression * >) const
  ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát
- Expression * copy () const
  dinamikusan foglalt memóriaterületen visszaadott másolat
- std::string show () const
  kifejezés megjelenítése std::string-ként

3.14.1. Részletes leírás

Valós számokat tároló kifejezés osztály.

3.14.2. Tagfüggvények dokumentációja

3.14.2.1. checkCyclic()

```
void NumberExpr::checkCyclic (
            std::vector< Expression * >  ) const  [inline], [virtual]
```

ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát

amennyiben igen, eval_error kivételt dob

Paraméterek

ps	- a kifejezésben korábban hivatkozott cellák, amire ismét hivatkozva körkörös hivatkozást kapunk

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

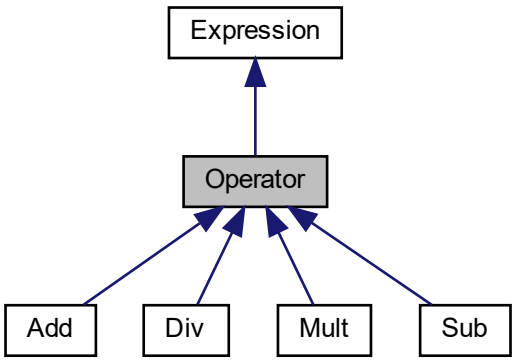
- srcs/expression_core.hpp

3.15. Operator osztályreferencia

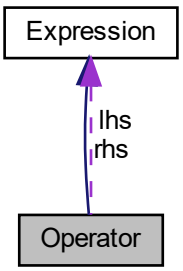
bináris műveleteket reprezentáló absztrakt osztály

`#include <expression.hpp>`

Az Operator osztály származási diagramja:



Az Operator osztály együttműködési diagramja:



Publikus tagfüggvények

- Operator (Expression ∗lhs, Expression ∗rhs)
  konstruktor
- Operator (const Operator &op)
  másoló konstruktor
- Operator & operator= (const Operator &op)
  értékadás operátor.
- void checkCyclic (std::vector< Expression ∗ > ps) const
  ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát
- void shift (int dx, int dy)
  rekurzívan minden hivatkozást adott oszlop- és sorszámmal eltol
- void relocate (Sheet ∗shp)
  a kifejezésben található hivatkozások célpontját áthelyezi egy másik számolótáblára
- virtual ∼Operator ()
  felszabadítja az operandusait

Statikus publikus tagfüggvények

- static Operator ∗ operandFromToken (Token_type tt, Expression ∗lhs, Expression ∗rhs)
  adott tokentípusnak megfelelő műveletet hoz létre

Védett attribútumok

- Expression ∗ lhs
  bal oldali operandus
- Expression ∗ rhs
  jobb oldali operandus

3.15.1. Részletes leírás

bináris műveleteket reprezentáló absztrakt osztály

3.15.2. Konstruktorok és destruktorok dokumentációja

3.15.2.1. Operator()

```
Operator::Operator (
            Expression * lhs,
            Expression * rhs )  [inline]
```

konstruktor

Paraméterek

lhs	- bal oldali operandus
rhs	- jobb oldali operandus

3.15.3. Tagfüggvények dokumentációja

3.15.3.1. checkCyclic()

```
void Operator::checkCyclic (
            std::vector< Expression * >  ) const  [inline], [virtual]
```

ellenőrzi, tartalmaz-e a kifejezés ciklikus referenciát

amennyiben igen, eval_error kivételt dob

Paraméterek

ps	- a kifejezésben korábban hivatkozott cellák, amire ismét hivatkozva körkörös hivatkozást kapunk

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/expression.hpp
- srcs/expression.cpp

3.16. Parser osztályreferencia

Kifejezéseket értelmező osztály.

```
#include <parser.hpp>
```

Publikus tagfüggvények

- Parser (std::string input)

  konstruktor: a megadott stringet tokenlistává alakítja
- Parser & operator= (const Parser &p)

  értékadó operátor.
- Parser (const Parser &p)

  másoló konstruktor
- void addToken (Token_type t)

  hozzáad a tokenek listájához egy megadott típusú tokent
- void addToken (std::string s)

  hozzáad a tokenek listájához egy STRING típusú adattokent a paraméterként megadott stringgel mint belső adat
- void addToken (double n)

  hozzáad a tokenek listájához egy NUMBER típusú adattokent a paraméterként megadott számmal mint belső adat
- void addTokenFromStr (std::string &str_buffer)

  megpróbál kiolvasni egy számot a kapott stringből, ha sikeresen, akkor hozzádja NUMBER tokenként, ha nem akkor STRING tokenként
- Expression ∗ parse (Sheet ∗shp=NULL)

  kifejezés értelmezése
- void parseTo (Sheet ∗shp, ExprPointer &ep)

  megpróbálja értelmezni a kifejezést az elejétől, ha sikertelen, syntax_error kivételt dob
- std::string show ()

  kiírja a tokenlistáját egy std::stringbe

3.16.1. Részletes leírás

Kifejezéseket értelmező osztály.

A parser osztályt egy std::string-el inicializáljuk, amit a Parser konstruktora tokenekre bont. Parser a parse tagfüggvényének segítségével megpróbálja értelmezni az adott kifejezést, és amennyiben ez lehetséges (azaz a kifejezés szintaktikailag helyes), létre is hozza a kifejezést dinamikus memóriaterületen, különben syntax_error típusú kivételt dob. A kifejezés értelmezése az alábbi szabályok alapján zajlik:
expression → factor (("-" | "+") factor)∗ ;
factor → unary (("/" | "∗") unary)∗ ;
unary → "-" unary | function | primary;
function → STRING "(" cell ":" cell ")";
cell → ('$')? STRING ('$' NUMBER)?;
primary → NUMBER | "(" expression ")" | cell;
Minden ilyen fent leírt szabályhoz tartozik egy-egy tagfüggvény, amelyeknek feladata, hogy a tokenlistának éppen aktív (current) tokenjétől kezdve megpróbáljon értelmezni egy megfelelő típusú kifejezést, ha ez lehetséges, akkor az ehhez szükséges tokeneket “elfogyasztja”, így a következő ilyen tagfüggvény azokat a tokeneket már nem fogja tudni felhasználni. Így láthatjuk, hogy a parse függvény igazából csak annyit tesz, hogy az első tokent állítja be aktív tokennek (current=0) és meghívja az expression-t értelmező függvényt. A kifejezések értelmezésének pontos menetének megvalósításához a program Robert Nystrom: Crafting Interpreters c. könyvéből merít ihletet, amely az alábbi linken elérhető: https://craftinginterpreters.com.

Megj.: az "-" unary mintára illeszkedő egy -1-el való szorzásra fordítja a parser, minden más mintának saját osztálya van

3.16.2. Tagfüggvények dokumentációja

3.16.2.1. parse()

```
Expression * Parser::parse (
             Sheet * shp = NULL )
```

kifejezés értelmezése

megpróbálja értelmezni a kifejezést az elejétől, ha sikertelen, syntax_error kivételt dob

Paraméterek

shp	- ha a kifejezés taralmaz referenciákat, akkor erre a táblára fognak vonatkozni

3.16.2.2. parseTo()

```
void Parser::parseTo (
             Sheet * shp,
             ExprPointer & ep )
```

megpróbálja értelmezni a kifejezést az elejétől, ha sikertelen, syntax_error kivételt dob

ha sikeres, akkor az adott tábla adott cellájába berakja a kifejezést

Paraméterek

shp	- ha a kifejezés taralmaz referenciákat, akkor erre a táblára fognak vonatkozni
ep	- az értelmezett kifejezés ebbe a cellába kerül

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/parser.hpp
- srcs/parser.cpp

3.17. Range osztályreferencia

Cellahivatkozások egy téglalap alakú tartományát reprezentáló osztály.

```
#include <expression.hpp>
```

Osztályok

- class iterator

  Tartományt sorfolytonosan bejáró iterátor.

Publikus tagfüggvények

- Range (CellRefExpr ∗bg, CellRefExpr ∗ed)

  konstruktor a tartomány két sarokcellájára mutató hivatkozás megadásával
- Range (const Range &r)

  másoló konstruktor
- Range & operator= (const Range &r)

  értékadás operátor.
- iterator begin () const

  tartomány első cellájára mutató iterátor visszaadása
- iterator end () const

  tartomány utolsó cellája utáni cellára mutató iterátor visszaadása
- std::string show () const

  tartomány megjelenítése std::string-ként "a1:c4" formátumban
- void shift (int dx, int dy)

  eltolja a taromány sarokcelláit adott sorral és oszloppal, amennyiben a sor/oszlop nem abszolút
- void relocate (Sheet ∗shp)

  a taromány sarokcelláinak célpontját áthelyezi egy másik számolótáblára
- ∼Range ()

3.17.1. Részletes leírás

Cellahivatkozások egy téglalap alakú tartományát reprezentáló osztály.

A téglalapot a bal felső és jobb alsó cellája határozza meg, és elvárás, hogy ez a két cella egy számolótáblán legyen.

3.17.2. Konstruktorok és destruktorok dokumentációja

3.17.2.1. Range()

```
Range::Range (
            CellRefExpr * bg,
            CellRefExpr * ed )
```

konstruktor a tartomány két sarokcellájára mutató hivatkozás megadásával

A két megadott sarokcella-hivatkozás bármely sorrendben, bármely átló szerint megadható, azonban a range mindig bal felső - jobb felső sorrendben tárolja el őket. Ezért a konstruktor új referenciákat készít és ezeket tárolja el végül és bejárni is a bal felső cellából kezdi így a bejárást.

3.17.2.2. ∼Range()

```
Range::~Range ( )  [inline]
```

sarokcella hivatkozások felszabadítása

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/expression.hpp
- srcs/expression.cpp

3.18. Sheet osztályreferencia

Számolótáblát reprezentáló osztály.

`#include <sheet.hpp>`

Publikus tagfüggvények

- Sheet ()
  konstruktor
- Sheet (const Sheet &)
- Sheet (size_t w, size_t h, double fill=0)
  konstruktor adott számmal inicializálással
- size_t getWidth () const
  tábla szélességének lekérdezése
- size_t getHeight () const
  tábla magasságának lekérdezése
- Sheet & operator= (const Sheet &)
  értékadó operátor.
- ExprPointer * operator[ ] (size_t i)
  adott sor lekérdezése 0-tól indexelve
- ExprPointer * parseCell (int col, int row) const
  tábla adott cellájára mutató pointer visszaadása oszlopszám és sorszám alapján
- ExprPointer * parseCell (std::string col, int row) const
  tábla adott cellájára mutató pointer visszaadása oszlopbetű és sorszám alapján
- bool checkRow (int r) const
  ellenőrzi, hogy a táblázatban szerepel-e adott sorszámú sor (1-től indexelve)
- bool checkCol (int col) const
  ellenőrzi, hogy a táblázatban szerepel-e adott sorszámú oszlop (1-től indexelve)
- int getYCoord (ExprPointer *cell) const
  visszaadja, hogy egy adott cellára mutató pointer melyik sorban van (0-tól indexelve)
- int getXCoord (ExprPointer *cell) const
- void copyTo (Sheet &sh) const
  átmásolja a tábla tartalmát egy másik, paraméterként kapott táblába.
- void resize (size_t w, size_t h, double fill=0)
  átméretezi a táblát.
- void formattedPrint (std::ostream &os=std::cout) const
  kiértékeli és kiírja a cellák értékét, illetve az oszlop és sorszámokat a kapott ostream-re
- void printValues (std::ostream &os=std::cout) const
  kiértékeli és kiírja a cellák értékét vesszővel elválasztva a kapott ostream-re
- void printExpr (std::ostream &os=std::cout) const
  kiírja a cellákban található kifejezéseket a kapott ostream-re
- ~Sheet ()
  felszabadítja a táblát

Statikus publikus tagfüggvények

- static int colNumber (std::string)
  oszlopbetű oszlopszámra alakítása (1-től indexelve)
- static std::string colLetter (int)
  oszlopszám oszlopbetűre alakítása (1-től indexelve)

3.18.1. Részletes leírás

Számolótáblát reprezentáló osztály.

A Sheet osztály egy N×M méretű dinamikus memóriaterületen sorfolytonosan tárolja el az adott cellában lévő kifejezés értékét az ExprPointer osztály pédényaiként.

3.18.2. Konstruktorok és destruktorok dokumentációja

3.18.2.1. Sheet() [1/2]

```
Sheet::Sheet (
            const Sheet & sh )
```

másoló konstruktor

3.18.2.2. Sheet() [2/2]

```
Sheet::Sheet (
            size_t w,
            size_t h,
            double fill = 0 )
```

konstruktor adott számmal inicializálással

Paraméterek

w	- létrehozandó tábla szélessége
h	- létrehozandó tábla magassága
fill	- a létrejövő tábla minden celláját ezzel a számmal inicializálja

3.18.3. Tagfüggvények dokumentációja

3.18.3.1. copyTo()

```
void Sheet::copyTo (
            Sheet & sh ) const
```

átmásolja a tábla tartalmát egy másik, paraméterként kapott táblába.

a paraméterként kapott tábla mérete nem változik meg, tehát ha kisebb volt, akkor az adatok részét nem másolja át, ha nagyobb volt, akkor a paraméterként kapott tábla fennmaradó részében az adatok változatlanok maradnak

3.18.3.2. getXCoord()

```
int Sheet::getXCoord (
            ExprPointer * cell ) const
```

visszaadja, hogy egy adott cellára mutató pointer melyik oszlopban van (0-tól indexelve)

3.18.3.3. resize()

```
void Sheet::resize (
            size_t w,
            size_t h,
            double fill = 0 )
```

átméretezi a táblát.

Ha kisebbre méretezzük a táblát, akkor az adatok egy része elveszik, ha nagyobbra, akkor a fill paraméterben megadott számmal tölti ki az újonnan keletkező részt

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

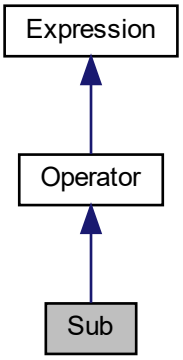
- srcs/sheet.hpp
- srcs/sheet.cpp

3.19. Sub osztályreferencia

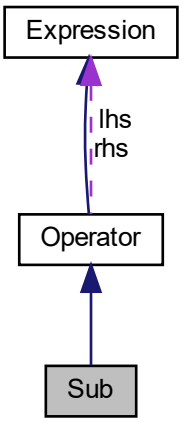
Kivonás műveletet reprezentáló osztály.

```
#include <expression.hpp>
```

A Sub osztály származási diagramja:



A Sub osztály együttműködési diagramja:



Publikus tagfüggvények

- **Sub** (Expression ∗lhs, Expression ∗rhs)
- double eval () const

  rekurzívan kiértékeli a kifejezést.
- std::string show () const

  kifejezés megjelenítése std::string-ként
- Expression ∗ copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.19.1. Részletes leírás

Kivonás műveletet reprezentáló osztály.

3.19.2. Tagfüggvények dokumentációja

3.19.2.1. eval()

```
double Sub::eval ( ) const  [inline], [virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

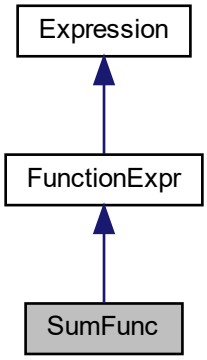
- srcs/expression.hpp

3.20. SumFunc osztályreferencia

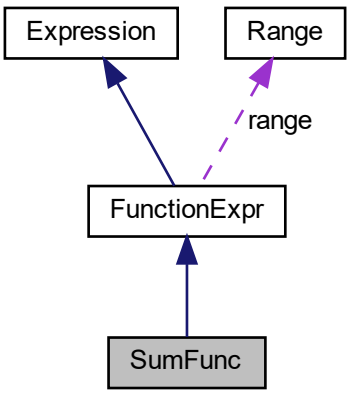
Tartományt összegző függvény osztály.

`#include <expression.hpp>`

A SumFunc osztály származási diagramja:



A SumFunc osztály együttműködési diagramja:



Publikus tagfüggvények

- **SumFunc** (Range r)
- **SumFunc** (CellRefExpr ∗topCell, CellRefExpr ∗bottomCell)
- double eval () const

rekurzívan kiértékeli a kifejezést.

- std::string show () const

  kifejezés megjelenítése std::string-ként

- Expression ∗ copy () const

  dinamikusan foglalt memóriaterületen visszaadott másolat

További örökölt tagok

3.20.1. Részletes leírás

Tartományt összegző függvény osztály.

3.20.2. Tagfüggvények dokumentációja

3.20.2.1. eval()

```
double SumFunc::eval ( ) const  [virtual]
```

rekurzívan kiértékeli a kifejezést.

kiértékelés közben eval_error típusa kivételt dobhat

Megvalósítja a következőket: Expression.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

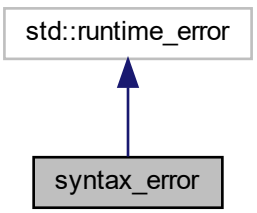
- srcs/expression.hpp
- srcs/expression.cpp

3.21. syntax_error osztályreferencia

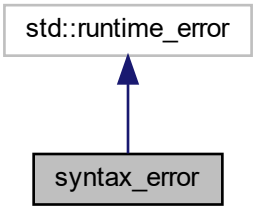
Saját exception osztály a szintaktikailag helytelen kifejezések értelmezésénél felmerülő hibákra.

```
#include <exceptions.hpp>
```

A syntax_error osztály származási diagramja:



A syntax_error osztály együttműködési diagramja:



3.21.1. Részletes leírás

Saját exception osztály a szintaktikailag helytelen kifejezések értelmezésénél felmerülő hibákra.

A szintakikai hiba részleteit a .what() tagfüggvényel kaphatjuk meg.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

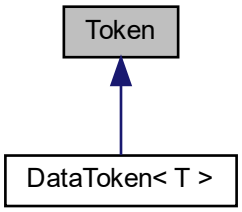
- srcs/exceptions.hpp

3.22. Token osztályreferencia

Tokenek osztálya: Az kifejezés értelmező ezen osztály példányaival tárolja el a kifejezéseket.

`#include <token.hpp>`

A Token osztály származási diagramja:



Publikus tagfüggvények

- Token ()
  alapértelmezett konstruktor
- Token (Token_type t)
  konstruktor típus megadásával
- Token_type getType () const
  típus lekérdezése
- std::string show () const
  token megjelenítése std::string-ként
- virtual Token ∗ copy ()
  dinamikusan foglalt memóriaterületen visszaadott másolat
- virtual ∼Token ()
  destruktor

Statikus publikus tagfüggvények

- static Token_type parseTokenType (char c)
  karakterhez megfelelő tokentípus rendelése

Védett attribútumok

- Token_type type
  token típusa

3.22.1. Részletes leírás

Tokenek osztálya: Az kifejezés értelmező ezen osztály példányaival tárolja el a kifejezéseket.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- srcs/token.hpp
- srcs/token.cpp

Tárgymutató