Trieda D2Variant

Popis

Trieda, ktorá reprezentuje variantný typ používaný pre parametre externých funkcií. Variantný typ môže by bu jednoduchá hodnota (logická hodnota, celé alebo reálne íslo, absolútny alebo relatívny as, text) alebo štrukturovaná hodnota (matica) jednoduchých hodnôt. Typ hodnoty variantného typu sa automaticky nastavuje poda typu hodnoty, ktorá sa nastavuje (metódy *setValue*), no pri pokuse o získanie hodnoty iného typu, aký variant obsahuje, je generovaná výnimka *std::logic_error*. Každá hodnota variantného typu obsahuje aj atribúty asová znaka, stavové príznaky a užívateské príznaky, podobne ako hodnoty ESL premenných.

Konštruktory

```
D2Variant (
        const D2Type type
= None
);
```

Predvolený konštruktor. Vytvorí variant jednoduchej hodnoty daného typu.

```
D2Variant (
        const int
rowsCount,
        const int
columnsCount
);
```

Konštruktor, ktorý vytvorí štrukturovaný variant s daným potom riadkov a stpcov.

```
D2Variant (
        const D2Variant&
variant
);
```

Kopírovací konštruktor. Vytvorí variant ako kópiu zdrojového variantu.

Operátory

```
D2Variant& operator= (
        const D2Variant&
variant
);
```

Operátor priradenia. Nastaví cieový variant kópiami hodnôt zdrojového variantu.

Metódy

```
void copyValue (
        const D2Variant&
source,
        const int
sourceRow,
        const int
sourceColumn,
        const int
destinationRow,
        const int
destinationColumn
);
```

Metóda skopíruje hodnotu (bunky) zdrojového variantu do (bunky) cieového variantu. V prípade použitia nad variantom jednoduchého typu indexy riadku aj stpca musia by 0. Pokus o prácu s bunkami štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
int getColumnsCount ()
const;
```

Vracia poet stpcov štrukturovaného variantu. Pre jednoduchý variant vracia 0.

Vracia stav nastavenosti užívateských príznakov daných bitovou maskou (bunky) variantu. Pokus o vrátenie stavu príznakov bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
bool getFlag (
        const unsigned
short flag,
        const int row = 0,
        const int column =
0
) const;
```

```
unsigned short getFlags (
        const int row = 0,
        const int column =
0
) const;
```

Vracia všetky užívateské príznaky (bunky) variantu. Pokus o vrátenie príznakov bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
int getRowsCount () const;
```

Vracia poet riadkov štrukturovaného variantu. Pre jednoduchý variant vracia 0.

```
D2Time getTimestamp (
        const int row = 0,
        const int column =
0
) const;
```

Vracia asovú znaku (bunky) variantu. Pokus o vrátenie asovej znaky bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
D2Type getType (
        const int row = 0,
        const int column =
0
) const;
```

Vracia typ (bunky) variantu. Pokus o vrátenie typu bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
bool getValueBoolean (
        const int row = 0,
        const int column =
0)
const;
```

Vracia hodnotu (bunky) variantu ako c++ typ *bool*. Typ (bunky) variantu musí by *Boolean*, inak je generovaná výnimka *std::logic_error*. Pokus o vrátenie hodnoty bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*. Ak hodnota nemá nastavený príznak platnosti, je generovaná výnimka *std::logic_error*.

```
D2Duration
getValueDuration (
        const int row = 0,
        const int column =
0
) const;
```

Vracia hodnotu (bunky) variantu ako c++ typ *D2Duration*. Typ (bunky) variantu musí by *Duration*, inak je generovaná výnimka *std::logic_error*. Pokus o vrátenie hodnoty bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*. Ak hodnota nemá nastavený príznak platnosti, je generovaná výnimka *std::logic_error*.

```
int getValueInteger (
        const int row = 0,
        const int column =
0
) const;
```

Vracia hodnotu (bunky) variantu ako c++ typ *int*. Typ (bunky) variantu musí by *Integer*, inak je generovaná výnimka *std::logic_error*. Pokus o vrátenie hodnoty bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*. Ak hodnota nemá nastavený príznak platnosti, je generovaná výnimka *std::logic_error*.

Vracia hodnotu (bunky) variantu ako c++ typ *double*. Typ (bunky) variantu musí by *Real*, inak je generovaná výnimka *std::logic_error*. Pokus o vrátenie hodnoty bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*. Ak hodnota nemá nastavený príznak platnosti, je generovaná výnimka *std::logic_error*.

```
double getValueReal (
        const int row = 0,
        const int column =
0
) const;
```

```
std::string getValueText (
        const int row = 0,
        const int column =
0
) const;
```

Vracia hodnotu (bunky) variantu ako c++ typ *std::string*. Typ (bunky) variantu musí by *Text*, inak je generovaná výnimka *std::logic_error*. Pokus o vrátenie hodnoty bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*. Ak hodnota nemá nastavený príznak platnosti, je generovaná výnimka *std::logic_error*. Vrátený text je v kódovaní UTF-8.

```
D2Time getValueTime (
        const int row = 0,
        const int column =
0
) const;
```

Vracia hodnotu (bunky) variantu ako c++ typ *D2Time*. Typ (bunky) variantu musí by *Time*, inak je generovaná výnimka *std::logic_error*. Pokus o vrátenie hodnoty bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*. Ak hodnota nemá nastavený príznak platnosti, je generovaná výnimka *std::logic_error*.

```
bool isSimple () const;
```

Vracia príznak i je variant jednoduchého typu (true) alebo štrukturovaný (false). Opak metódy *D2Variant::isStructured*.

```
bool isStructured () const;
```

Vracia príznak i je variant štrukturovaný (true) alebo jednoduchý (false). Opak metódy *D2Variant::isSimple*.

```
bool isValid (
        const int row = 0,
        const int column =
0
) const;
```

Vracia príznak platnosti (bunky) variantu. Pokus o zistenie stavu bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
bool isWeak (
        const int row = 0,
        const int column =
0
) const;
```

Vracia stav nastavenosti príznaku *Weak* (bunky) variantu. Pokus o zistenie stavu bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
void setFlag (
        const unsigned
short flag,
        const bool set,
        const int row = 0,
        const int column =
0
);
```

Nastaví alebo vynuluje zvolené užívateské príznaky (bunky) variantu na základe bitovej masky príznakov. Pokus o nastavenie bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
void setFlags (
        const unsigned
short flags,
        const int row = 0,
        const int column =
0
);
```

Nastaví všetky užívateské príznaky (bunky) variantu. Pokus o nastavenie bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
void setInvalid (
        const int row = 0,
        const int column =
0
);
```

Vynuluje príznak platnosti (bunky) variantu - zneplatní hodnotu. Pokus o nastavenie bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

```
void setRowsCount (
        const int count
);
```

Zmení rozmer štukturovaného variantu na daný poet riadkov. Hodnoty existujúcich riadkov zostanú zachované. V prípade volania metódy nad neštrukturovaným variantom je generovaná výnimka *std::logic_error*.

```
void setSimple (
        const D2Type type
);
```

Zmení typ variantu na jednoduchý a hodnotu nastaví na predvolenú. Ak sa nový typ zhoduje s existujúcim, metóda nerobí ni.

```
void setStructured (
        const int
rowsCount,
        const int
columnsCount
);
```

Zmení typ variantu na štrukturovaný s daným potom riadkov a stpcov. Ak sa nezmenil poet stpcov, tak sa metóda správa rovnako ako *D2Variant::setRowsCount*;

```
void setTimestamp (
        const D2Time
timestamp,
        const int row = 0,
        const int column =
0
);
```

Nastaví asovú znaku (bunky) variantu. Pokus o nastavenie bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

Metódy nastavia hodnotu jednoduchého variantu alebo bunky štrukturovaného variantu poda vstupného typu. Zárove nastavia stav hodnoty na platnú, vynulujú užívateské príznaky a nastavia asovú znaku hodnoty na aktuálny as. Pokus o nastavenie bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.

Text musí by v UTF-8 kódovaní.

```
void setValue (
        const bool value,
        const int row = 0,
        const int column =
0,
        const D2Time
timestamp = D2Time::now ()
);
void setValue (
        const int value,
        const int row = 0,
        const int column =
0,
        const D2Time
timestamp = D2Time::now ()
);
void setValue (
        const double value,
        const int row = 0,
        const int column =
0,
        const D2Time
timestamp = D2Time::now ()
);
void setValue (
        const D2Duration
value,
        const int row = 0,
        const int column =
0,
        const D2Time
timestamp = D2Time::now ()
);
void setValue (
        const D2Time value,
        const int row = 0,
        const int column =
0,
        const D2Time
timestamp = D2Time::now ()
);
void setValue (
        const std::string
value,
        const int row = 0,
        const int column =
0,
        const D2Time
timestamp = D2Time::now ()
);
```

```
void setWeak (
        const bool weak =
true,
        const int row = 0,
        const int column =
0
);
```

Nastaví alebo vynuluje príznak (bunky) variantu *Weak*. Pokus o nastavenie bunky štrukturovaného variantu mimo rozsahu generuje výnimku *std::out_of_range*.