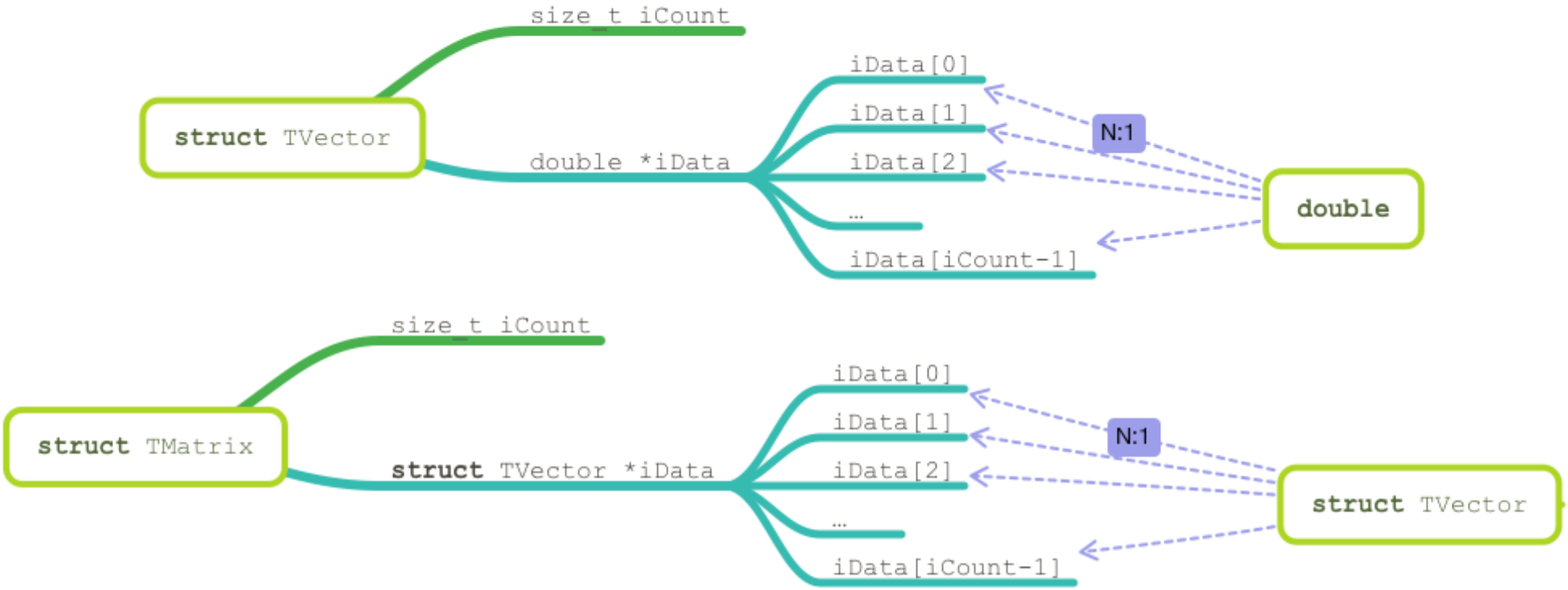
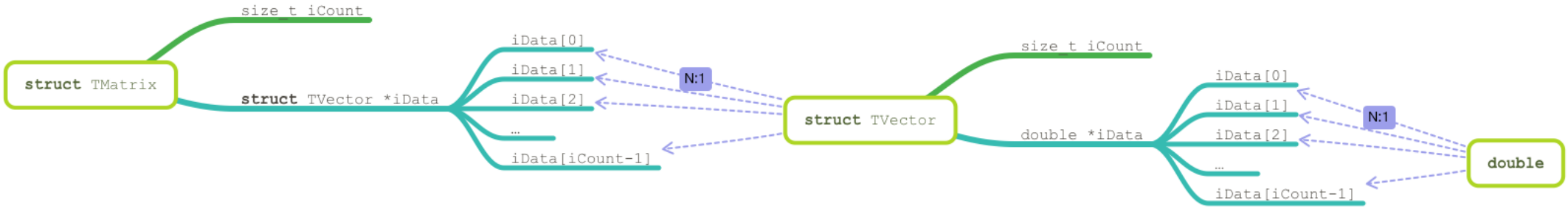
Struktury a dynamická paměť

Petyovský, Macho, Richter (bpc2a_cv12), ver. 2017.2

Definujte strukturu `TVector` pro dynamické pole dat typu `double`.
Definujete strukturu `TMatrix`, která bude obsahovat dynamické pole struktur `TVector`. Struktury `TVector` , budou nést pole dat, která si můžeme představit jako řádky matice, která však nemusí mít řádky stejně dlouhé. Jelikož je počet prvků struktury `TMatrix` dynamický, je možné měnit i počet řádků.
Pomocí těchto struktur realizujte program, který načte řádky ze souboru a vytvoří jejich součet.

Výsledný pohled: jedna struktura `TMatrix` obsahuje pole struktur `TVector`, které obsahují pole dat typu `double`. V každé struktuře je proměnná `iCount`, která nese informaci o počtu prvků ve struktuře (viz interaktivní odkaz).





*Pozn.: Při realizaci můžete samostatně postupovat podle následujících bodů, nebo si [stáhnout projekt](), kde jsou již některé **počáteční body zadání vyřešeny**.*

0 Založte projekt a v něm vytvořte dva zdrojové soubory: (`main.c` , `TVector.c`) a hlavičkový soubor (`TVector.h`). V prvním zdrojovém souboru bude funkce `main`, ve druhém budou funkce pracující se strukturou. Pro kontrolu správnosti práce se soubory a pamětí použijte knihovnu `check`.

1 V hlavičkovém souboru proveďte deklaraci struktury `TVector`, která obsahuje proměnnou `iData` typu ukazatel na `double`. Tento ukazatel bude obsahovat počáteční adresu pole které může mít libovolnou délku (dynamicky alokované). Pro délku pole nadefinujte proměnnou `size_t iCount`.

2 Do souboru s funkcí `main` vložte příkaz pro načtení hlavičkového souboru `TVector.h` a ve funkci `main` nadefinujte proměnnou `vect` typu `TVector` a inicializujte její vnitřní proměnné na nulové hodnoty.

3 Napište funkci `VectorCreate(struct TVector *aVector, size_t aCount)`, která alokuje data pro `aCount` hodnot typu `double` a nastaví vnitřní proměnné struktury na kterou ukazuje `aVector`.
Ve funkci `main` pomocí volání funkce `VectorCreate(struct TVector *aVector, size_t aCount)`

připravte proměnnou `vect` na uložení deseti hodnot. Hodnoty pole v proměnné `vect` nastavte p od nuly do devíti (nejprve přímo pomocí operátoru indexace pole a následně pomocí funkce `VectorSetAt`).

4 Napište funkci pro tisk hodnot struktury `TVector`. Funkce bude mít prototyp:
`enum TErrCode VectorStore(const struct TVector* const aVector, FILE *aOutp)`
a vytiskne hodnoty dat struktury `TVector` ve formátu:

`pocet: Hodnota0 Hodnota1 … HodnotaN`

Pro tisk počtu prvku vektoru využijte (`"%zu:"`), hodnoty tiskněte formátovaně na dvě desetinná místa pomocí: (`" %.2lf"`).
Pro testování využijte na místě druhého parametru stream pro konzolu `stdout`.
Pozn.: Soustřeďte se na vlastní činnost funkce. Po dokončení ostatních bodů funkci doplňte o testování vstupních proměnných a jako návratovou hodnotu pro typ chyby použijte typ a hodnoty z `enum TErrCode` ze vzorového projektu.

5 Pomocí funkce `VectorStore` vytiskněte strukturu z bodu 3 na konzolu.

6 Napište funkci `VectorDestroy(struct TVector *aVector)`, která odalokuje data ve struktuře a nastaví proměnné tak, aby signalizovaly, že ve struktuře nejsou platná data: `{.iCount=0, .iData=NULL}`
Strukturu z předchozích bodů uvolněte pomocí funkce `VectorDestroy(struct TVector *aVector);`

7 Ve funkci `main` otevřete textový soubor pro čtení (*`data.in`*).

V souboru pro čtení budou vstupní data ve formátu:

```
počet řádků dat
počet hodnot na řádku: hodnota hodnota … hodnota
…
počet hodnot na řádku: hodnota hodnota … hodnota
```

Např.:

```
4
3: 3.15 21 546
6: 33.5 5.15 21 54 6 -3
4: 3.15 221 46 -15
6: 33.5 5.15 21 54 6 -3
```

8 Struktura `TVector` bude sloužit k uložení jednoho pole, tj. jednoho řádku ze souboru.
Napište funkci `VectorLoad(struct TVector *aVector, FILE *aInput)`, která načte jeden řádek souboru do struktury `TVector`. Funkce bude mít jako parametr otevřený vstupní soubor `FILE *aInput` a ukazatel na proměnnou reprezentující vektor `struct TVector *aVector`.
Funkce načte první hodnotu na řádku, naalokuje pole pro data. V případě úspěšného přečtení dat ze souboru uloží funkce do struktury nový počet prvků i adresu počátku nově alokovaného pole (zajistěte korektní dealokaci předchozí alokované paměti).
Návratové hodnoty:

- `0` v pořádku
- `1` špatně vstupní parametr (`NULL`)
- `2` nepodařilo se naalokovat
- `3` chyba v souboru (formát vstupních dat)

Pozn.: Soustřeďte se na vlastní činnost funkce. Po dokončení ostatních bodů funkci doplňte o testování vstupních proměnných a jako návratovou hodnotu pro typ chyby použijte typ a hodnoty z `enum TErrCode` *ze vzorového projektu.*

9 Nyní budeme pracovat v souborech (`TMatrix.h, TMatrix.c`). Napište strukturu `TMatrix`, která bude obsahovat ukazatel na pole struktur `TVector` a počet prvků pole. Struktura bude obsahovat proměnnou `iData` reprezentující ukazatel na pole struktur `struct TVector` a hodnotu `iCount` udávající počet prvků pole (typ `size_t`).

10 Ve funkci `main` vytvořte proměnnou s názvem `mat` typu `struct TMatrix`. Vytvořenou proměnnou inicializujte na literálovou hodnotu: `{.iCount=0, .iData=NULL}`

11 Napište funkci `MatrixLoad(struct TMatrix *aMatrix, FILE *aInput)`, která načte postupně řádky souboru do struktury `TMatrix`. Funkce načte první řádek souboru, kde je hodnota udávající počet řádků v souboru tj. i počet hodnot/řádků ve struktuře. V předané struktuře naalokuje datové prvky pro načítání řádků a nastaví proměnnou pro počet hodnot ve struktuře. Pomocí funkce `VectorLoad` postupně načte hodnoty/pole řádků.
Návratové hodnoty:

- `0` v pořádku
- `1` špatně vstupní parametr (`NULL`)
- `2` nepodařilo se naalokovat
- `3` chyba v souboru (formát vstupních dat)

Pozn.: Soustřeďte se na vlastní činnost funkce. Po dokončení ostatních bodů funkci doplňte o testování vstupních proměnných a jako návratovou hodnotu pro typ chyby použijte typ a hodnoty z `enum TErrCode` *ze vzorového projektu.*

12 Napište funkci `MatrixStore(const struct TMatrix* const aMatrix, FILE *aOutp)`, která vytiskne do souboru data ze struktury `TMatrix`. Na první řádek vytiskne hodnotu počtu řádků a následně data pro jednotlivé řádky pomocí funkce `VectorStore`. Pro tisk se použije stejný formát jako je ve vstupním souboru.
Pozn.: Soustřeďte se na vlastní činnost funkce. Po dokončení ostatních bodů funkci doplňte o testování vstupních proměnných a jako návratovou hodnotu pro typ chyby použijte typ a hodnoty z `enum TErrCode` *ze vzorového projektu.*

13 Napište funkci `MatrixSummarize(struct TMatrix* const aMatrix)`, která přidá do struktury `TMatrix` jeden řádek na poslední pozici (o délce nejdelšího pole ze stávajících struktur) a do něj uloží součet polí z ostatních struktur/řádků. Pokud nejsou v poli hodnoty (je kratší), potom se tyto hodnoty nepřičítají (považují se za nulové).
Pozn.: Soustřeďte se na vlastní činnost funkce. Po dokončení ostatních bodů funkci doplňte o testování vstupních proměnných a jako návratovou hodnotu pro typ chyby použijte typ a hodnoty z `enum TErrCode` *ze vzorového projektu.*

14 Následně ji naplňte ze vstupního otevřeného souboru pomocí funkce `MatrixLoad`. Strukturu zobrazte pomocí `MatrixStore` na `stdout`. Pomocí volání funkce `MatrixSummarize` přidejte na poslední řádek struktury součet ostatních řádků. Výslednou strukturu z bodu 13 vytiskněte na `stdout` pomocí funkce `MatrixStore`.

15 Napište funkci `MatrixDestroy`, která korektně uvolní paměť struktury `TMatrix`. a proměnné struktury nastaví do nulových hodnot.
Pozn.: Soustřeďte se na vlastní činnost funkce. Po dokončení ostatních bodů funkci doplňte o testování

vstupních proměnných a jako návratovou hodnotu pro typ chyby použijte typ a hodnoty z `enum TErrCode` *ze vzorového projektu*.

16 Korektně ukončete funkci `main` a vraťte hodnotu `0`. V případě, že během vykonávání funkce `main` dojde k chybě, program korektně ukončete a vraťte hodnotu `1`.