Dieselgate 2

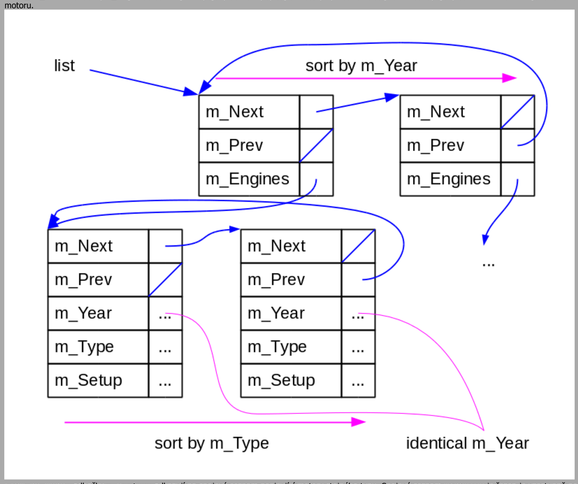
Termín odevzdání: **22.12.2019 23:59:59**
Pozdní odevzdání s penalizací: **06.01.2020 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení: **5.5000**
Max. hodnocení: **5.0000** (bez bonusů)
Odevzdaná řešení: 11 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy: 1 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je realizovat funkce (ne celý program, pouze funkce), které dokáží spravovat archiv s nastavením řídicí jednotky spalovacího motoru.

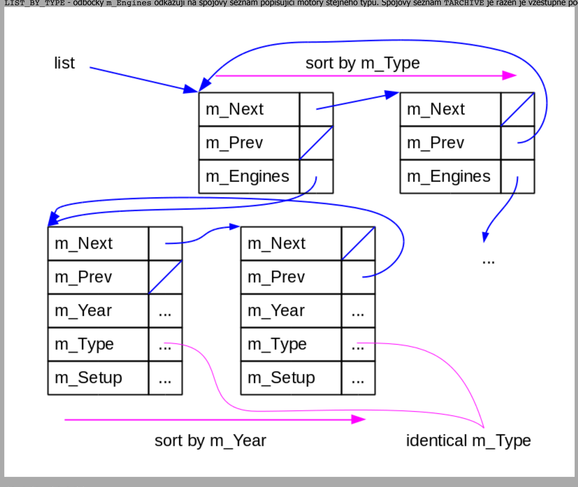
V aféře "dieselgate" vystupuje jako důležitý důkaz nastavení řídicí jednotky motoru. Nastavení se liší podle typu motoru a roku výroby. Vlastní nastavení pak tvoří sada čísel, která je hlavně velmi tajná (a pro implementaci tohoto úkolu ani nebude zase tak důležitá). Vaším úkolem je realizovat sadu funkcí, které dokáží s takovými nastaveními pracovat.

Nastavení chceme ukládat v podobě spojových seznamů - viz obrázek. Struktura `TARCHIVE` tvoří dvojsměrný spojový seznam (ukazatel `m_Next` odkazuje na další prvek v seznamu, `m_Prev` na předchozí), odbočky (`m_Engines`) pak odkazují na spojové struktury s popisem jednotlivých motorů. Celý archiv je organizovaný jedním z následujících způsobů:

- `LIST_BY_YEAR` - odbočky `m_Engines` odkazují na spojový seznam popisující motory vyráběné ve stejném roce. Spojový seznam `TARCHIVE` je řazen vzestupně podle roku. Každý spojový seznam `TENGINE` je pak řazen vzestupně podle typu motoru.



- `LIST_BY_TYPE` - odbočky `m_Engines` odkazují na spojový seznam popisující motory stejného typu. Spojový seznam `TARCHIVE` je řazen je vzestupně podle typu. Každý spojový seznam `TENGINE` je pak řazen vzestupně podle roku.



Rozhraní funkcí je následující:

```
#define LIST_BY_YEAR       0
#define LIST_BY_TYPE       1

#define TYPE_MAX           100
#define SETUP_MAX          100

typedef struct TEngine
 {
   struct TEngine * m_Next;
   struct TEngine * m_Prev;
   int              m_Year;
   char             m_Type  [ TYPE_MAX ];
   int              m_Setup [ SETUP_MAX ];
 } TENGINE;

typedef struct TArchive
 {
   struct TArchive * m_Next;
   struct TArchive * m_Prev;
   TENGINE         * m_Engines;
 } TARCHIVE;

TARCHIVE  * AddEngine      ( TARCHIVE        * list,
                             int               listBy,
                             TENGINE         * engine );
void        DelArchive     ( TARCHIVE        * list );
TARCHIVE * ReorderArchive  ( TARCHIVE        * list,
                             int               listBy );
```

TENGINE
: struktura popisuje jeden motor. Složka `m_Year` udává rok, složka `m_Type` je typ motoru (ASCIIZ řetězec). Složka `m_Next` je odkazem na další motor ve stejné kategorii nebo má hodnotu `NULL` pro poslední prvek spojového seznamu. Složka `m_Prev` je odkazem na předešlý motor ve stejné kategorii nebo má hodnotu `NULL` pro první prvek spojového seznamu. Složka `m_Setup` je vyplněna informacemi o nastavení řídicí jednotky. Vaše implementace tuto složku nepotřebuje, proto do ní nebude nijak zasahovat. Vzhledem k tajnému charakteru nastavení je dokonce zakázáno čtení a kopírování (tedy je potřeba pracovat s existujícími strukturami `TENGINE` a nevytvářet nové).

TARCHIVE
: je pomocná struktura propojující seznamy `TENGINE`. Složka `m_Next` odkazuje na další prvek `TARCHIVE` (NULL pro poslední v seznamu), složka `m_Prev` odkazuje na předchozí prvek `TARCHIVE` (NULL pro první v seznamu), složka `m_Engines` obsahuje ukazatel na spojový seznam motorů se stejnou charakteristikou (typ/rok).

`AddEngine(list, listby, engine)`
: dostane v parametru `list` odkaz na existující archiv nastavení řídicích jednotek. Jejím úkolem je zařadit nový motor. Odkaz na dynamicky alokovanou strukturu s vyplněnými údaji o motoru je předaný v parameteru `engine`. Parametr `listby` má hodnotu LIST_BY_YEAR / LIST_BY_TYPE a udává, způsob organizace archivu (viz výše). Funkce použije alokovanou strukturu motoru a začlení ji do existujícího archivu. Je zakázáno strukturu kopírovat, je potřeba pouze správně proházet odkazy. Funkce vrátí ukazatel na první prvek v archivu po provedení změny.

`DelArchive(list)`
: funkce uvolní dynamicky alokovanou paměť, kterou používaly struktury v existujícím archivu `list`.

`ReorderArchive(list, listBy)`
: funkce změní uspořádání existujícího archivu `list` tak, aby odpovídalo uspořádání `listBy` (LIST_BY_YEAR / LIST_BY_TYPE). Funkce musí zachovat existující struktury `TENGINE` (pouze přehazuje odkazy), může podle potřeby alokovat a uvolňovat struktury `TARCHIVE`. Návratovou hodnotou funkce je ukazatel na první prvek archivu po provedení úprav.

Odevzdávejte zdrojový soubor, který obsahuje implementaci požadovaných funkcí `AddEngine`, `DelArchive` a `ReorderArchive`. Do zdrojového souboru přidejte i další Vaše podpůrné funkce, které jsou z nich volané. Funkce bude volaná z testovacího prostředí, je proto důležité přesně dodržet zadané rozhraní funkce. Za základ pro implementaci použijte kód z přiloženého archivu. V kódu chybí vyplnit těla funkcí a případné další podpůrné funkce. Ukázka obsahuje testovací funkci `main`, uvedené hodnoty jsou použité při základním testu. Všimněte si, že vkládání hlavičkových souborů, deklarace struktur a funkce `main` jsou zabalené v bloku podmíněného překladu (`#ifdef/#endif`). Prosím, ponechte bloky podmíněného překladu i v odevzdávaném zdrojovém souboru. Podmíněný překlad Vám zjednoduší práci. Při kompilaci na Vašem počítači můžete program normálně spouštět a testovat. Při kompilaci na Progtestu funkce `main` a vkládání hlavičkových souborů "zmizí", tedy nebude kolidovat s hlavičkovými soubory a funkcí `main` testovacího prostředí.

Váš program bude spouštěn v omezeném testovacím prostředí. Je omezen dobou běhu (limit je vidět v logu referenčního řešení) a dále je omezena i velikost dostupné paměti. Očekává se základní řešení, úloha má cvičit práci se spojovými seznamy.

Nápověda:

- Zkopírujte si ukázku z přílohy a použijte ji jako základ Vašeho řešení.
- Do funkce `main` si můžete doplnit i další Vaše testy, případně ji můžete libovolně změnit. Důležité je zachovat podmíněný překlad.
- Testovací prostředí pozná pokusy o kopírování struktur `TENGINE` a takové řešení odmítne.

Vzorová data: Download

Referenční řešení

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test meznich hodnot': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.013 s (limit: 2.000 s)
    - CPU time: 0.016 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty + test prace s pameti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.003 s (limit: 4.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 1.00 * (5.00 + 0.50) = 5.50

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**6**	--	--	--
Řádek kódu:	**124**	20.67 ± 9.69	34	`archiveByType`
Cyklomatická složitost:	**26**	4.33 ± 2.62	8	`archiveByYear`

11 **07.12.2019 08:03:07** Download

Stav odevzdání: Ohodnoceno
Hodnocení: 5.5000

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test meznich hodnot': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.018 s (limit: 2.000 s)
    - CPU time: 0.020 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty + test prace s pameti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.005 s (limit: 4.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00)
- Použité nápovědy: 1
- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 1.00 * (5.00 + 0.50) = 5.50

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**6**	--	--	--
Řádek kódu:	**170**	28.33 ± 17.37	50	`AddEngine`
Cyklomatická složitost:	**40**	6.67 ± 4.61	15	`AddEngine`