Zaawansowane metody programowania obiektowego

Zadanie 1

Alokacja i dealokacja

autor: Michał Przewoźniczek
konsultacja: Marcin Komarnicki

Wrocław, 13.09.2018

Cel zadania

Oprogramować klasę `CTable`, pozwalającą na testowanie działania konstruktorów.

Klasa `CTable` ma posiadać niepubliczne pole `s_name` przechowujące wartość typu `string` (w zależności od preferencji można użyć dowolnego typu: `CString`, `std::string`, `QString`, lub analogicznego). Klasa ma również przechowywać dynamicznie alokowaną tablicę typu `int*`.

W momencie utworzenia obiektu przez konstruktor bezparametrowy tablicy ma zostać nadana domyślna długość (np. 10), którą następnie będzie można zmienić.

Obiekty klasy `CTable` mają pozwalać na zrealizowanie następujących operacji:

- Zmianę długości tablicy
- Przypisanie określonej komórce tablicy wartości typu `int`
- Odczyt wartości określonej komórki
- Utworzenie klonu obiektu (innego obiektu posiadającego te same wartości w tablicy)
- Przypisanie obiektowi A, wartości i stanu tabeli w obiekcie B (po wykonaniu takiej operacji w obiekcie A tabela ma posiadać tą samą długość i te same wartości, co tablica w obiekcie B)
- Zwrócenie informacji o obiekcie do zmiennej typu string w formacie: `(<nazwa obiektu> len: <liczba pozycji> values: <wszystkie wartości z tablicy oddzielone przecinkami>)`

Uwaga: Niektóre powyższe operacje mogą spowodować błędy, lub się nie powieść. W takiej sytuacji obiekt powinien zwrócić informację o błędzie. W niniejszym programie zakazane jest użycie wyjątków. W związku z powyższym informacja o powodzeniu operacji powinna być przekazywana jako wynik funkcji, lub poprzez jeden z parametrów funkcji/metody (Np.: `iGetElement(int iOffset, int *piSucc)`, może zwracać wartość danej pozycji w tablicy, a poprzez parametr `*piSucc` zwracać informację, czy pobranie zakończyło się sukcesem, lub `bGetElement(int iOffset, int *piElemValue)` może zwracać informację o sukcesie/porażce jako wynik działania funkcji, a wartość danej pozycji w tablicy poprzez parametr).

Klasa `CTable` musi posiadać następujące konstruktory chrakteryzujące się następującym działaniem:

- bezparametrowy: `CTable()`
  - przypisuje polu `s_name` domyślną wartość (proszę pamiętać o użyciu odpowiednich stałych)
  - wypisuje na ekran tekst „bezp: '<s_name>'", gdzie <s_name> oznacza wartość pola `s_name`
- z parametrem: `CTable(string sName, int iTableLen)`
  - przypisuje polu `s_name`, wartość `sName`
  - wypisuje na ekran tekst: „parametr: '<s_name>'"
- kopiujący: `CTable(CTable &pcOther)`
  - przypisuje polu `s_name`, wartość `pcOther.s_name` i doklejający tekst „_copy". Na przykład, jeśli `pcOther.s_name` = „test" to wartość pola `s_name` dla obiektu utworzonego konstruktorem kopiującym będzie: „test_copy"
  - wypisuje na ekran tekst: kopiuj: „<s_name>'"

Ponadto klasa ma posiadać:

- **Destruktor**, wypisujący na ekran następujący tekst: „usuwam: '<s_name>'"
- **Metodę**, `void vSetName(string sName)`, przypisującą polu `s_name`, wartość `sName`

Program musi posiadać tekstowy interfejs użytkownika, który będzie pozwalał na:

- Dynamiczne utworzenie dowolnej liczby obiektów typu `CTable`
- Określenie długości tablicy dla dowolnego z utworzonych dynamicznie obiektów `CTable`
- Skasowanie dowolnego dynamicznie utworzonego obiektu typu `CTable`
- Skasowanie wszystkich dynamicznie utworzonych obiektów typu `CTable`
- Nadanie nowej nazwy dowolnemu z dynamicznie utworzonych obiektów `CTable`
- Sklonowanie dowolnego dynamicznie utworzonego obiektu `CTable` i dodanie klona do listy/puli dynamicznie utworzonych obiektów klasy `CTable`
- Wypisanie na ekran dowolnego dynamicznie utworzonego obiektu `CTable` (należy użyć metody zwracającej stan obiektu `CTable` w zmiennej typu `string`)
- Umożliwienie wpisania wartości dowolnej komórki, wybranego dynamicznie utworzonego obiektu `CTable`
- Program ma być odporny na błędy użytkownika (np. w przypadku, gdy wskaże on obiekt `CTable` spoza dostępnego zakresu)
- Dynamicznie tworzone obiekty `CTable` można przechowywać w dowolny sposób (np. w tablicy lub w wektorze). Musi on jednak zapewniać możliwość zdefiniowania dowolnej liczby obiektów
- Należy zwrócić uwagę na to, że wszystkie dynamicznie utworzone obiekty powinny być skasowane, po zakończeniu działania programu