Opracowanie: dr inż. Łukasz Maliński, Politechnika Śląska, 2022

LABORATORIUM 1.3
DYNAMICZNA ALOKACJA PAMIĘCI

Zadanie podstawowe [2,0 pkt.]

■ Napisz funkcję alokującą tablicę instancji struktur wybranego przez siebie typu.

Wejście: rozmiar tablicy,

Wyjście: adres alokacji.

Samodzielnie zaproponuj typ strukturalny o 2 polach (różnego typu). Tablica w trakcie alokacji nie musi być inicjalizowana.

Wskazówki: Jedyna trudność w tym poleceniu to dobranie typu zwracanego wskaźnika. Dobierze pola tak aby łatwo było Ci wygenerować dane testowe.

■ Napisz funkcję zwalniającą pamięć dla tablicy tworzonej powyżej.

Wejście: adres alokacji,

Wyjście: brak. Funkcja powinna jednak uziemić przekazany jej wskaźnik.

Wskazówki: Jak wyżej trudność leży w dobrze typu wskaźnika.

■ Napisz funkcję, która przyjmie dwie niezależne tablice instancji (typ jak w poprzednich poleceniach) i zwróci nową tablicę zwierającą tylko instancje, które wystąpiły w obu tablicach jednocześnie.

Wejście: dwie tablice instancji,

Wyjście: adres nowej alokacji, rozmiar nowej tablicy.

Tablica wyjściowa powinna zawierać tylko unikatowe instancje – nie powinno w niej być dwóch instancji o identycznej treści. Kolejność instancji nie ma znaczenia, ale rozmiar tablicy wyjściowej musi być dokładnie dopasowany do zawartości.

Wskazówki: W tym poleceniu są dwie zasadnicze trudności. Pierwsza to jak zwrócić rozmiar tablicy i adres alokacji równocześnie. Jest na to kilka sposobów, ze zwracaniem struktury jako wyniku włącznie.

Druga to napisanie algorytmu wyszukującego identyczne instancje tak, aby w rezultacie otrzymać zbiór tylko unikatowych instancji. Zauważ, że dwie lub więcej identycznych instancji może także wystąpić w jednej z tablic wejściowych. Problem można rozwiązać dwuetapowo

1) Stworzyć tablicę która zwiera wszystkie instancje, które pojawiły się choć raz w obu tablicach.

2) Oczyścić tą tablicę z duplikatów. Co będzie się wiązało z realokacją pamięci. Realokację omówiono na wykładzie – trzeba tylko podmienić typ danych.

Sortowanie instancji może ułatwić rozwianie tego problemu, ale wtedy trzeba zdefiniować jak sortować instancje struktur – co nam mówi, że jedna instancja powinna wystąpić wcześniej niż inna.

■ Zademonstruj działanie wszystkich funkcji w programie głównym.

Wskazówki: Napisanie funkcji do wypełniania treścią instancji, lub nawet całej tablicy (losowymi danymi) znacznie ułatwi prezentację działania.

Ocenianie:

Funkcja alokująca tablicę instancji:	0,2 pkt.
Funkcja zwalniająca pamięć:	0,4 pkt.
Funkcja wyszukująca unikaty:	0,8 pkt.
Prezentacja działania w programie głównym:	0,4 pkt.
Ogólna jakość kodu:	0,2 pkt.

Opracowanie: dr inż. Łukasz Maliński, Politechnika Śląska, 2022

Zadanie ambitne [2,0 pkt.]:

▲ Napisz funkcję alokującą tablicę 2D typu int o dowolnych zadanych wymiarach.

Wejście: wymiary tablicy,

Wyjście: adres alokacji.

Funkcja powinna samodzielnie wybierać typ alokacji na bazie rozmiaru jaki tablica zajmie w pamięci. Jeśli docelowy rozmiar jest mniejszy niż 1 MiB (mebibajt), to alokacja powinna zachodzić w sposób ciągły. W innym wypadku powinna zachodzić w sposób fragmentaryczny. Funkcja nie powinna zwracać informacji jakiego typu była alokacja.

Wskazówki: Rozmiar docelowy da się obliczyć znając zdaną liczbę elementów i rozmiar w bajtach typu int. Sprawdź w Internecie ile bajtów ma mebibajt, nie pomyl tego z 1 MB (megabajtem).

■ Funkcja powinna na bieżąco (po każdej alokacji cząstkowej) weryfikować, czy alokacja przebiegła pomyślnie i dać możliwość odwrócenia procesu alokacji, jeśli na dowolnym etapie nie uda się zaalokować pamięci. Innymi słowy, jeśli w trakcie tworzenia alokacji, na dowolnym etapie zabraknie pamięci, to funkcja powinna zwolnić całą pamięć, którą już udało się jej zaalokować – wrócić do stanu z przed jej działania. Jest to bardzo ważne, aby uniknąć masowego wycieku pamięci.

Wskazówki: Użyj do sygnalizacji przez wartość (std::nothrow), aby uprościć sobie pracę, gdyż obsługa wyjątków może doprowadzić tu do bardzo skomplikowanego kodu.

Zauważ, że przy alokacji ciągłej sprawa jest w zasadzie prosta, bo bez względu na rozmiary, alokacja następuje zawsze tylko dwa razy (tablica wskaźników, tablica danych). Problem zatem może wystąpić tylko przy alokacji tablicy danych. Jeśli ta się nie powiedzie, to trzeba tylko zwolnić wcześniej zaalokowaną tablicę wskaźników. Jeśli tablica danych zostanie poprawnie zaalokowana, to nic już nie powinno pójść źle. Partycjonowanie nie ma wpływu na zużycie pamięci.

*Sprawa znacząco się komplikuje przy alokacji fragmentarycznej, gdyż tam liczba alokacji to **n** + 1 gdzie **n** to liczba wierszy. Oznacza to, że przy każdej alokacji pojedynczego wiersza, może zabraknąć pamięci. W takim wypadku trzeba najpierw zwolnić wszystkie zalokowane dotychczas wiersze, a potem dopiero tablicę wskaźników. Kluczowym aspektem jest tu wykombinowanie jak rejestrować ile wierszy już się udało zaalokować, gdyż absolutnie nie możemy zwolnić więcej niż zaalokowaliśmy!*

■ Jeśli posiadasz wykonane zadanie domowe (nr 3) z tego tematu, to w jednym programie zademonstruj działanie funkcji alokującej (z tego zadana) i zwalniającej (z zadania domowego). Stosując komunikaty diagnostyczne (cerr) sprawdź, czy funkcje te prawidłowo wybierają/identyfikują typ alokacji i czy prawidłowo rozpoznają wymiary tablicy (jeśli zrealizujesz wariant trudniejszy).

Wskazówki: Koniecznie użyj komunikatów cerr aby zaznaczyć w kodzie, że to są informacje diagnostyczne, a nie docelowe działanie programu. Sprawdzając, wypisz funkcji alokującej wybrany przez nią typ alokacji, wymiary tablicy i rozmiar alokacji w bajtach. W funkcji zwalniającej wypisz zidentyfikowany typ alokacji i zidentyfikowane wymiary (jeśli realizujesz wariant trudniejszy).

Ocenianie:

Automatyczny wybór typu alokacji:	0,2 pkt.
Implementacja obu alokacji:	0,4 pkt.
Przywrócenie stanu pamięci przy błędzie:	
dla alokacji ciągłej:	0,2 pkt.
dla alokacji fragmentarycznej:	0,8 pkt.
Prezentacja działania w programie głównym:	0,2 pkt.
Ogólna jakość kodu:	0,2 pkt.