laboratorium: zadanie 7 termin: 4–7 maja 2021 r.

KURS JĘZYKA C++

TABLICA LICZB ZMIENNOPOZYCYJNYCH

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Prolog.

Resource acquisition is initialization (pozyskiwanie zasobu poprzez inicjalizację), w skrócie RAII, to popularny wzorzec projektowy w języku C++. Technika ta jest realizowana za pomocą obiektów opakowujących zasoby (ang. *wrapper*) i łączy przyjęcie określonego zasobu z konstrukcją opakowania (inicjalizacja za pomocą konstruktora) a zwolnienie zasobu z destrukcją opakowania (automatycznie uruchamiany destruktor przed likwidacją obiektu). Ponieważ zagwarantowane jest automatyczne wywołanie destruktora, gdy zmienna opuszcza zasięg swojej deklaacji, to w konsekwencji zasób zostanie zwolniony w momencie, gdy skończy się czas życia zmiennej opakowującej zasób — stanie się tak również w przypadku zgłoszenia wyjątku (podczas zwijania stosu).

Techika RAII jest kluczową koncepcją przy pisaniu kodu odpornego na błędy. Mariusz Jaskólka na swoim blogu programistycznym pisze tak:

> Bardzo popularne dziś podejście do zarządzania pamięcią, jakim jest obowiązkowe użycie odśmiecacza pamięci (ang. *Garbage Collector*) działającego w tle w języku takim jak C++ nie ma miejsca, ponieważ wiąże się z narzutem wydajnościowym, co z kolei kłóci się z koncepcją *Zero Overhead Principle*. Dodatkowo mechanizm ten dotyczy tylko jednego rodzaju zasobu — pamięci. I choć jest to zasób wykorzystywany przez programy najczęściej, to istnieją jednak inne, które również wymagają starannego ich zwalniania i mechanizmów, które to ułatwiają.

Zadanie 1.

Zdefiniuj klasę opakowującą dla tablicy liczb zmiennopozycyjnych `double[]` zgodnie ze wzorcem RAII. Tablica o zadanym rozmiarze ma być utworzona na stercie podczas pracy konstruktora a usunięta ze sterty w destruktorze. Klasa opakowująca `tab_dbl` powinna implementować semantykę kopiowania i przenoszenia.

```
class tab_dbl
{
    double *tab; // tablica liczb zmiennopozycyjnych
    int dl; // rozmiar tablicy
public:
    explicit tab_dbl(int rozm); // wyzerowana tablica liczb
    tab_dbl(const tab_dbl &t); // konstruktor kopiujący
```

```
    tab_dbl(tab_dbl &&t); // konstruktor przenoszący
    tab_dbl& operator = (const tab_dbl &t); // przypisanie kopiujące
    tab_dbl& operator = (tab_dbl &&t); // przypisanie przenoszące
    ~tab_dbl(); // destruktor
// ...
};
```

Definicję klasy umieść w przestrzeni nazw `obliczenia`.

Użyj jednolitej inicjalizacji do wyzerowania tablicy na początku jej istnienia. Zgłoś wyjątek `invalid_argument`, gdy przekazany do konstruktora rozmiar tablicy nie będzie liczbą dodatnią.

W klasie opakowującej zdefiniuj także operatory indeksowania, zwracające odpowiednio wartość dla tablic stałych i referencję do komórki w przypadku tablic modyfikowalnych. Gdy indeks będzie miał wartość spoza dopuszczalnego zakresu zgłoś wyjątek `out_of_range`.

Rzetelnie przetestuj całą funkcjonalność opakowania na tablicę ze szczególnym uwzględnieniem zgłaszanych wyjątków.

Zadanie 2.

Uzupełnij definicję opakowania `tab_dbl` o konstruktor bezargumentowy, który utworzy tablicę liczb zmiennopozycyjnych o maksymalnym możliwym rozmiarze, będącym potęgą 2. Wykorzystaj operator `new` z parametrem `nothrow`. Konstruktor ten powinien zainicjalizować utworzoną tablicę losowymi wartościami z zakresu $[0, 1)$.

Tablicę o jakim rozmiarze udało Ci się utworzyć?

Zadanie 3.

Uzupełnij definicję opakowania `tab_dbl` o konstruktor, który utworzy i zainicjalizuje tablicę w oparciu o listę wartośći `initializer_list<double>`.

Następnie zdefiniuj funkcję (albo operator mnożenia), która będzie liczyć iloczyn skalarny dwóch tablic o identycznych rozmiarach. Wstaw do funkcji asercję sprawdzającą, czy długości obu tablic są takie same.

Czy asercja zadziałała, gdy podałeś tablice o różnych rozmiarach? Jak wyłączyć asercje?

Istotne elementy programu.

- Podział programu na pliki nagłówkowe i pliki źródłowe (wyodrębniony osobny plik z funkcją `main()` z testami).
- Użycie przestrzeni nazw `obliczenia`.
- Implementacja semantyki kopiowania i przenoszenia.
- Operatory indeksowania.
- Zgłaszanie wyjątków i ich wyłapywanie w testach.
- Posługiwanie się asercjami.