laboratorium: zadanie 3 termin: 20–24 marca 2023 r.

KURS JĘZYKA C++

ZMIENNA Z HISTORIĄ ZMIAN WARTOŚCI

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Prolog.

Przeglądarki przechowują historię przeglądanych przez nas stron internetowych. Funkcjonalność ta umożliwia szybki powrót do strony, którą ostatnio oglądaliśmy; możemy wycofać się do tylu stron wstecz, ile mamy zapisanych w historii.

Pamiętanie historii odwiedzanych stron ma i dobre i złe strony: zaletą tego mechanizmu jest możliwość uzupełniania adresów URL podczas ich wpisania, wadą może być inwigilacja naszych poczynań w Internecie.

Zadanie.

Zdefiniuj klasę `liczba`, która będzie opakowaniem dla zmiennej typu `double`. Opakowanie to ma dodatkowo przechowywać historię zmian wartości zmiennopozycyjnych przechowywanych w zmiennej oraz umożliwiać przywracanie wartości poprzednich pobranych z historii.

Maksymalna długość historii ma zostać określona za pomocą pewnej stałej statycznej w klasie `liczba` (do celów testowych ustaw jej wartość na niskim poziomie, na przykład 3). Samą historię przechowywanych w zmiennej wartości zaimplementuj na tablicy utworzonej dynamicznie na stercie operatorem `new[]` (tablica ta ma być utworzona w konstruktorze, ale pamiętaj by w destruktorze zwolnić przydzieloną pamięć operatorem `delete[]`). Jeśli historia przekracza zdefiniowaną długość, to początkowe wpisy mają być bezpowrotnie tracone.

Przywracanie wartości zmiennej z historii ma być oparte na strategii LIFO (stos) — bieżąca wartość jest zastępowana wartością odczytaną z bufora historii (zawsze odczytujemy najpóźniej wpisaną wartość). Zadbaj o to aby historia zmiennej nigdy nie była pusta: w buforze na dane historyczne musi znajdować się co najmniej jedna wartość (wybraną komórkę z bufora można wykorzystać do pamiętania bieżącej wartości zmiennej, bez dodatkowego pola w klasie). Przywracanie wartości z historii z pojedynczą wartością w buforze (jest to więc bieżąca wartość zmiennej) nie powinna powodować żadnej zmiany w obiekcie `liczba`.

Postaraj się zaimplementować operacje na historii optymalnie — bez względu na długość bufora opracje wstawienia nowej wartości do zmiennej i przywrócenia starej wartości z historii powinny wykonywać się w stałym czasie, niezależnym od długości tablicy przechowującej dane historyczne. Programując bufor wykorzysaj zawinięcie w tablicy (następny po ostatnim elemencie jest pierwszy element w tablicy), co przyspieszy działanie tej struktury — oprócz samej tablicy na dane historyczne pamiętaj więc miejsce poczętkowe i ilość wartości historycznych.

Implementacja zmiennej z danymi historycznymi typu `liczba` ma posiadać cztery konstruktory: konstruktor z zadaną wartością, konstruktor bezparametrowy i jednocześnie delegatowy

(przekazana do poprzedniego konstruktora wartość to 0), konstruktor kopiujący (pobierana jest tylko bieżąca wartość ze wzorca bez historii) i przenoszący. Aby uzupełnić semantykę kopiowania i przenoszenia zdefiniuj opowiednie operatory przypisania (przypisanie kopiujące i przenoszące), przy czym każde przypisanie ma skutkować automatycznym uzupełnianiem historii zmiennej.

Napisz też program rzetelnie testujący działanie zaimplementowanej zmiennej typu `liczba`.

Istotne elementy programu.

- Podział programu na plik nagłówkowy (np. `liczba.hpp`) z definicją klasy `liczba` reprezentującej zmienną z danymi historycznymi, plik źródłowy (np. `liczba.cpp`) z definicjami funkcji składowych dla zmiennej oraz plik źródłowy z funkcją `main()` (np. `main.cpp`).
- Obiekt zmiennej ma być inicjalizowany na kilka różnych sposobów: konkretną wartością, domyślnie (konstruktor delegatowy) za pomocą 0, przez skopiowanie z innej zmiennej (konstruktor kopiujący), za pomocą przeniesienia zawartości ze zmiennej tymczasowej (konstruktor przenoszący).
- Obiekt zmiennej z historią ma być kopiowalny (przypisanie kopiujące i przenoszące).
- W funkcji `main()` należy przetestować działanie zaimplementowanej zmiennej z danymi historycznymi typu `liczba`.