Zadanie 6

Otwarto: poniedziałek, 16 grudnia 2024, 00:00
Wymagane do: środa, 15 stycznia 2025, 23:59

Liczniki

Wstęp

W standardzie C++20 wprowadzono do języka C++ moduły. Mimo że w dostępnych kompilatorach ich implementacja nie jest kompletna, a w niektórych aspektach eksperymentalna, proponujemy zapoznanie się z tym nowym rozszerzeniem języka. Celem tego zadania jest poznanie:

- podstaw używania modułów w C++,
- paradygmatu programowania obiektowego w C++.

Definicja licznika, rodzaje liczników

Licznik jest układem cyfrowym zliczającym impulsy pojawiające się na jego wejściu. Licznik ma dzielnik wstępny o wartości `<P>`. Po każdym zliczonym impulsie licznik pomija kolejnych `<P>` impulsów. Licznik zgłasza zdarzenia po osiągnięciu określonych wartości. Wartości licznika oraz wartość parametru `<P>` są liczbami całkowitymi z przedziału od `0` do `UINT64_MAX`.

Mamy kilka typów liczników:

- Licznik modulo zlicza od zera do wartości maksymalnej `<M>`, po czym ponownie od zera. Ten licznik zgłasza zdarzenie przepełnienia, gdy jego wartość zmienia się z `<M>` na zero. Jeśli wartość maksymalna jest równa zeru, to licznik ma zawsze wartość zero i zgłasza przepełnienie przy każdym zliczonym impulsie.
- Licznik Fibonacciego zlicza od zera do `UINT64_MAX`, po czym się zatrzymuje. Ten licznik zgłasza zdarzenie, gdy jego wartość osiąga wartość będącą liczbą Fibonacciego, czyli 1, 2, 3, 5, 8 itd.
- Licznik geometryczno-dziesiętny zlicza w kolejnych cyklach od `0` do `9`, od `0` do `99`, od `0` do `999`, ..., od `0` do `999999999999`, po czym znów od `0` do `9` itd. Ten licznik zgłasza zdarzenie przepełnienia, gdy jego wartość zmienia się z wartości maksymalnej na zero.

Polecenie

Napisać program obsługujący zbiór liczników. Program czyta dane ze standardowego wejścia. Poprawna linia z danymi wejściowymi ma jeden z następujących formatów:

- `M <C> <P> <M>`

  Tworzy nowy licznik modulo o numerze `<C>` z parametrami `<P>`, `<M>`.
- `F <C> <P>`

  Tworzy nowy licznik Fibonacciego o numerze `<C>` z parametrem `<P>`.
- `G <C> <P>`

  Tworzy nowy licznik geometryczno-dziesiętny o numerze `<C>` z parametrem `<P>`.
- `D <C>`

  Usuwa licznik o numerze `<C>`.
- `P <C>`

  Wypisuje literę `C`, spację, numer licznika, spację, wartość tego licznika i znak nowej linii.
- `A <T>`

?

Wysyła do wszystkich liczników `<T>` impulsów. Powoduje wypisanie na standardowe wyjście zdarzeń wygenerowanych przez wszystkie liczniki od poprzedniego wykonania tego polecenia lub od początku działania programu, jeśli to polecenie jest wykonywane pierwszy raz. Każde zdarzenie wypisywane jest w osobnej linii zakończonej znakiem nowej linii i składa się z litery `E`, spacji, numeru licznika `<C>` zgłaszającego to zdarzenie, spacji i liczby impulsów od poprzedniego wykonania tego polecenia lub od początku działania programu, jeśli to polecenie jest wykonywane pierwszy raz. Zdarzenia wypisywane są chronologiczne według ich czasu zgłoszenia. Zdarzenia zgłoszone w tym samym czasie są sortowane rosnąco w kolejności numerów liczników je zgłaszających. Jeśli nie ma zdarzeń, to niczego nie wypisuje.

Początkowa wartość nowego licznika wynosi zero. Parametry `<C>`, `<P>`, `<M>`, `<T>` są liczbami całkowitymi z przedziału od `0` do `UINT64_MAX`.

Nazwa polecenia i jego parametry oddzielone są pojedynczą spacją. Linie wejściowe są rozdzielone znakiem nowej linii. Po ostatniej linii znak nowej linii jest opcjonalny. Poprawne dane wejściowe nie zawierają innych białych znaków niż wymienione powyżej.

Obsługa błędów

Program na bieżąco sprawdza, czy dane wejściowe nie zawierają błędów. Dla każdej błędnej linii program wypisuje na standardowe wyjście diagnostyczne komunikat

```
ERROR L
```

gdzie `L` oznacza numer linii. Linie są numerowane od 1. Komunikat o błędzie kończy się znakiem nowej linii. Program ignoruje zawartość błędnych linii.

Jako błędną należy potraktować linię, w której jest próba utworzenia nowego licznika, jeśli licznik o podanym numerze już istnieje, lub próba usunięcia nieistniejącego licznika.

Wymagania formalne

Program powinien kończyć się kodem 0.

Implementacja powinna korzystać z paradygmatu programowania obiektowego.

Implementacja powinna być podzielona na kilka modułów. Każdy moduł powinien znajdować się w osobnym pliku z rozszerzeniem `.cppm`.

Nie wolno korzystać z poleceń preprocesora zaczynających się znakiem `#`.

Należy dostarczyć skrypt `makefile` lub `Makefile` umożliwiającej skompilowanie programu za pomocą polecenia `make`. Wywołanie tego polecenia bez parametrów powinno tworzyć plik wykonywalny `counters`. Skrypt powinien zawierać cele `.PHONY` i `clean`. Może też zawierać inne cele, np. `test`. Skrypt powinien zawierać reguły kompilowania przyrostowego i opisywać zależności między plikami.

Jako rozwiązanie należy wstawić w Moodle wspomniane wyżej pliki. Nie wolno wstawiać innych plików.

Wskazówki dotyczące kompilowania

Polecenie kompilujące wstępnie plik nagłówkowy `vector`:

```
clang++ -std=c++20 -O2 -Wall -Wextra -Wno-experimental-header-units -Wno-pragma-system-header-outside-header -xc++-system-header --precompile vector -o vector.pch
```

Polecenie kompilujące wstępnie moduł `module1` zawarty w pliku `module1.cppm` i zależny od modułów `vector` oraz `module2` (zakładamy, że w bieżącym katalogu są pliki `vector.pch` i `module2.pcm`):

```
clang++ -std=c++20 -O2 -Wall -Wextra -Wno-experimental-header-units -fprebuilt-module-path=. --precompile -fmodule-file=vector.pch module1.ccpm -o module1.pcm
```

Polecenie kompilujące moduł `module1`:

```
clang++ -std=c++20 -O2 -Wall -Wextra -fprebuilt-module-path=. -c module1.pcm -o module1.o
```

Polecenie linkujące moduły `module1` i `module2`:

```
clang++ module1.o module2.o -o executable
```

?

Rozwiązanie będzie kompilowane i testowane na maszynie students i tamże w powyższych poleceniach jako kompilatora `clang++` należy użyć kompilatora `/opt/llvm/19.1.4/bin/clang++`.

Ocenianie rozwiązania

Rozwiązanie niespełniające wymagań formalnych lub niekompilujące się może dostać zero punktów.

Ocena automatyczna

Za testy automatyczne zostanie przyznana ocena z przedziału od 0 do 6 punków. Za ostrzeżenia wypisywane przez kompilator zostanie odjęty 1 punkt. Nie ma punktów ułamkowych.

Ocena jakości kodu

Ocena jakości kodu jest z przedziału od 0 do 4 punktów. Nie ma punktów ułamkowych. Odejmujemy punkty za:

- złe formatowanie kodu (niepoprawne wstawianie spacji, wcięć, odstępów, magiczne stałe itd.);
- niedostateczne komentarze;
- zbędne pliki;
- niedostateczny podział na moduły;
- niezastosowanie się do paradygmatu programowania obiektowego:
  - brak wirtualnych destruktorów (tam gdzie trzeba), jeśli nie zostanie wykryte przez testy automatyczne;
  - zbyt skomplikowane klasy, brak wydzielenia klas pomocniczych;
  - zależności cykliczne typów;
  - brak właściwej enkapsulacji, m.in. zwracanie modyfikowalnych struktur danych, zbyt duże możliwości modyfikowania stanu obiektu (zbyt dużo publicznych metod, „głupie” publiczne settery lub gettery);
  - instrukcja `switch` zamiast wykorzystania polimorfizmu;
  - zbyt duże klasy lub metody;
  - zbyt dużo powiązań między klasami;
  - powtórzenia kodu;
  - rzutowanie w dół, tego NIE powinno być;
  - niewłaściwe użycie słów kluczowych `public`, `protected`, `private`, `virtual`, `override`, `final`;
- problemy z zarządzaniem pamięcią, niekorzystanie ze sprytnych wskaźników;
- użycie typu zmiennoprzecinkowego lub obliczeń wielokrotnej precyzji;
- niedostateczny `makefile` lub `Makefile`:
  - brak reguły `.PHONY` lub `clean`;
  - wiele szczegółowych reguł zamiast reguły ogólnej;
  - braki w opisie zależności między plikami;
  - braki w kompilowaniu przyrostowym;
- niezastosowanie się do wymagań i uwag sformułowanych w poprzednich zadaniach, np. braki w użyciu słów kluczowych `const` i `noexcept`.

Przykład użycia

Przykład użycia jest częścią specyfikacji. Formatowanie wypisywanych danych powinno być dokładnie zgodne z formatowaniem w podanym przykładzie użycia. Przykład użycia znajduje się w załączonych w plikach:

	counters_test_0.err	13 grudnia 2024, 21:23
	counters_test_0.in	13 grudnia 2024, 21:23
	counters_test_0.out	13 grudnia 2024, 21:23
	counters_tests.zip	16 stycznia 2025, 17:07