Zadanie 1: Arytmetyka

Otwarto: poniedziałek, 16 października 2023, 08:00
Wymagane do: środa, 8 listopada 2023, 16:00

Arytmetyka przybliżonych wartości

Tam gdzie dokonujemy pomiarów wielkości fizycznych, wyniki są obarczone pewnym błędem, np. 5m ± 10%. Każdą taką przybliżoną wartość traktujemy jak zbiór możliwych wartości. Zaimplementuj pakiet operacji arytmetycznych na takich przybliżonych wartościach zgodny z załączonym interfejsem `ary.h`. W szczególności, pakiet ma zawierać:

- definicję typu `struct wartosc`,
- konstruktory:
  - `wartosc_dokladnosc(x, p)` zwraca x ± p% (dla p > 0),
  - `wartosc_od_do(x, y)` zwraca (x+y)/2 ± (y-x)/2 (dla x < y),
  - `wartosc_dokladna(x)` zwraca x ± 0
- selektory:
  - `in_wartosc(x, y)` ⇔ wartość x może być równa y,
  - `min_wartosc(x)` = kres dolny możliwych wartości x (lub -∞ jeśli możliwe wartości x nie są ograniczone od dołu, lub nan jeśli x jest puste),
  - `max_wartosc(x)` = kres górny możliwych wartości x (lub ∞ jeśli możliwe wartości x nie są ograniczone od góry, lub nan jeśli x jest puste),
  - `sr_wartosc(x)` = średnia (arytmetyczna) wartości `min_wartosc(x)` i `max_wartosc(x)` (lub nan jeśli `min_wartosc(x)` i `max_wartosc(x)` nie są skończone),
- modyfikatory:
  - `plus(a, b)` = { x + y : `in_wartosc(a, x)` ∧ `in_wartosc(b, y)` },
  - `minus(a, b)` = { x - y : `in_wartosc(a, x)` ∧ `in_wartosc(b, y)` },
  - `razy(a, b)` = { x · y : `in_wartosc(a, x)` ∧ `in_wartosc(b, y)` },
  - `podzielic(a,b)` = {x/y: `in_wartosc(a, x)` ∧ `in_wartosc(b, y)` }.

Zakładamy przy tym implicite, że wszystkie argumenty typu `double` są liczbami rzeczywistymi (tzn. są różne od `HUGE_VAL`, `-HUGE_VAL` i `NAN`).
Natomiast w przypadku, gdy wynik nie jest liczbą rzeczywistą, powinien być odpowiednią z wartości: `HUGE_VAL`, `-HUGE_VAL` lub `NAN`.

Rozwiązując to zadanie możesz przyjąć następujące zasady ułatwiające rozumowanie:

- Przyjmij, że modyfikatory domykają wynikowe zbiory wartości – to znaczy, jeżeli wynikiem jest przedział otwarty, to przyjmij, że zostaje on zamieniony na przedział domknięty.
- Operacje na wartościach przybliżonych są monotoniczne ze względu na zawieranie się zbiorów możliwych wartości. To znaczy, jeżeli wartości przybliżone x, y i z spełniają, jako zbiory możliwych wartości, x ⊆ y, to:

  `plus(x, z)` ⊆ `plus(y, z)`

  `plus(z, x)` ⊆ `plus(z, y)`

  i podobnie dla innych operacji arytmetycznych.
- Kilka przykładów opartych o powyższą zasadę możesz znaleźć w pliku `przyklad.c`. Komenda kompilacji (pliki `ary.c` i `ary.h` muszą być w katalogu):
  `gcc @opcje przyklad.c ary.c -o przyklad.e -lm`
  Opcja `-lm` załącza `math.h`. Koniecznie musi znajdować się na końcu komendy kompilacji.
- Liczby zmiennopozycyjne i operacje na nich potrafią być zaskakujące. Na przykład, standard IEEE przewiduje dwie

?

reprezentacje zera (`+0.0` i `-0.0`), przy czym `1.0 /. 0.0 = HUGE_VAL`, oraz `1.0 /. (-0.0) = -HUGE_VAL`.
Może być to pomocne, np. jeśli dzielisz przez wartość przybliżoną, która zawiera jednostronne otoczenie zera.
Ale może też okazać się pułapką, gdy rozważasz dzielenie przez wartość dokładnie równą zero.
Pamiętaj, że w definicji operacji `podzielic` występuje dzielenie "matematyczne", które nie jest określone gdy dzielimy przez zero.
Możesz przyjąć, że liczba typu `double` jest równa zeru, jeśli poniższa funkcja daje dla niej wynik `true`:

```
bool iszero(double x) { return fabs(x) < 1e-10; }
```

Twoje rozwiązanie ma być umieszczone w pliku o nazwie `ary.c` i pasować do specyfikacji interfejsu `ary.h`. Należy również załączyć plik `ary.h` uzupełniony jedynie o definicję typu `struct wartosc`.

ary.h	16 października 2023, 14:21
opcje	17 września 2023, 13:54
przyklad.c	24 października 2023, 11:08