

Zadanie C

Podprogramy zbiorowe

Punktów do uzyskania: 5

Generalia

Zadanie polega na implementacji podprogramów obsługi zbiorów w oparciu o poniższe założenia:

- Uniwersum obejmuje liczby całkowite poczynszyszy od 1 do 4095 włącznie.
- Zbiór jest implementowany w postaci tablicy typu **int**, przy czym:
 - Elementy w tablicy się nie powtarzają.
 - Zbiór *n*-elementowy zajmuje *n* pierwszych elementów tablicy, zaś wartość elementu o indeksie *n* zawsze wynosi -1.

Użyte dalej słowo *zbiór* oznacza zbiór w rozumieniu opisanej implementacji.

Opis podprogramów

- **Procedura Add**
Dodaje liczbę całkowitą do zbioru, o ile należy do uniwersum. Przewiduje argumenty:
 1. Dowolnej wartości typu **int**.
 2. Odniesienia do zbioru przyjmującego element.
- **Procedura Create**
Z tablicy dowolnych wartości typu **int** tworzy zbiór, przewidując argumenty:
 1. Nieujemnej liczby całkowitej typu **int** określającej używaną ilość elementów tablicy danej drugim argumentem.
 2. Odniesienie do tablicy dowolnych liczb typu **int** stanowiących źródło dla elementów tworzonego zbioru.
 3. Odniesienie do tworzonego zbioru.
- **Procedura Complement**
Wyznacza dopełnienie zbioru, przewidując argumenty odniesień do zbiorów:
 1. Dopełnianego.
 2. Dopełnianego.
- **Procedura Union**
Wyznacza sumę zbiorów, przewidując argumenty odniesień do zbiorów:
 1. Pierwszego operanda.
 2. Drugiego operanda.
 3. Wynikowej sumy.
- **Procedura Intersection**
Wyznacza część wspólną zbiorów, przewidując argumenty odniesień do zbiorów:
 1. Pierwszego operanda.
 2. Drugiego operanda.
 3. Wynikowego przecięcia.

- **Procedura Difference**
Wyznacza różnicę zbiorów, przewidując argumenty odniesień do zbiorów:
 1. Odjemnej.
 2. Odjemnika.
 3. Wynikowej różnicy.
- **Procedura Symmetric**
Wyznacza różnicę symetryczną zbiorów, przewidując argumenty odniesień do zbiorów:
 1. Pierwszego operanda.
 2. Drugiego operanda.
 3. Wynikowej różnicy symetrycznej.
- **Funkcja logiczna Subset**
Określa zawieranie zbiorów, przewidując argumenty odniesień do zbiorów:
 1. Ewentualnie zawieranego.
 2. Ewentualnie zawierającego.
- **Funkcja logiczna Equal**
Określa równość zbiorów, przewidując dwa argumenty odniesień do zbiorów.
- **Funkcja logiczna Empty**
Określa pustość zbioru przekazanego argumentem.
- **Funkcja logiczna Nonempty**
Określa niepustość zbioru przekazanego argumentem.
- **Funkcja logiczna Element**
Określa przynależność elementu do zbioru, przewidując argumenty:
 1. Dowolnej liczby typu **int** stanowiącej ewentualny element zbioru.
 2. Odniesienie do zbioru ewentualnie zawierającego element.
- **Funkcja Arithmetic typu double**,
Zwraca wartość średniej arytmetycznej elementów zbioru danego argumentem. Przyjmujemy, że średnia arytmetyczna zbioru pustego wynosi 0.
- **Funkcja Harmonic typu double**,
Zwraca wartość średniej harmonicznego elementów zbioru danego argumentem. Przyjmujemy, że średnia harmoniczna zbioru pustego wynosi 1.
- **Procedura MinMax**
Wyznacza minimalny i maksymalny element zbioru, przewidując argumenty:
 1. Odniesienia do zbioru.
 2. Wskaźnika do zmiennej przyjmującej wartość elementu minimalnego.
 3. Referencji do zmiennej przyjmującej wartość elementu maksymalnego.Dla zbioru pustego wartości minimum i maksimum nie są zmieniane względem podanych przy wywołaniu.

- **Procedura Cardinality**
Określa moc zbioru przewidując argumenty:
 1. Odniesienia do zbioru.
 2. Wskaźnika do zmiennej przyjmującej moc zbioru.
- **Procedura Properties**
Wyznacza informacje o zbiorze na podstawie zadanego ciągu znakowego. Przewiduje argumenty:
 1. Odniesienia do zbioru
 2. Ciągu znakowego określającego operacje do wykonania i zawierającej wyłącznie znaki ze zbioru o opisanym poniżej znaczeniu:
 - **a** - wyznaczenie średniej arytmetycznej
 - **h** - wyznaczenie średniej harmonicznego
 - **m** - wyznaczenie minimalnego i maksymalnego elementu zbioru
 - **c** – wyznaczenie mocy zbioru.
 3. Referencji do zmiennej typu **double** przechowującej wartość średniej arytmetycznej elementów zbioru.
 4. Wskaźnika do zmiennej typu **double** przechowującej wartość średniej harmonicznego elementów zbioru.
 5. Referencji do zmiennej całkowitej przechowującej wartość elementu minimalnego.
 6. Wskaźnika do zmiennej całkowitej przechowującej wartość elementu maksymalnego.
 7. Referencji do zmiennej przechowującej moc zbioru.

Dodatkowe uwarunkowania

- **Zawartości tablic określających wejściowe argumenty nie mogą ulec zmianie.**
- **Tablice określające wyniki działań są zawsze długości minimalnej dla poprawnej odpowiedzi. Dlatego odwołania do elementów spoza koniecznego zakresu mogą prowadzić do błędów.**
- **Elementy tablic reprezentujące zbiory wynikowe muszą być posortowane niemalejąco.**
- Plik z rozwiązaniem musi nosić nazwę `source.cpp` i być spakowany programem *Zip*.
- Pierwsza linia kodu źródłowego musi zawierać imię i nazwisko autora rozwiązania.
- Jedynym dozwolonym do włączenia plikiem nagłówkowym jest plik `io-stream`.
- W całym kodzie źródłowym dopuszczalna jest wyłącznie wiedza przedstawiona na wykładzie, w szczególności zabronione jest używanie:
 - Zmiennych globalnych.
 - Sekwencji `string`, **struct**, **class**, **new** oraz **delete**.
 - Własnych tablic.
 - Identyfikatorów własnych podprogramów rozpoczynających się znakiem podkreślenia.
 - Typu wskaźnikowego oraz typu referencyjnego poza odniesieniami do argumentów podprogramów
- **Jakakolwiek próba obejścia powyższych warunków skutkuje dyskwalifikacją rozwiązania i wyzerowaniem punktacji po upływie terminu oddania zadania.**