Zajęcia 20

Temat: Zawody 2. Powtórzenie i podsumowanie.

Czas trwania: 2x45 min

Cel zajęć:

projektuje i programuje proste problemy z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, tablice dwuwymiarowe, rekurencję, pisze własne funkcje rekurencyjne, struktury danych, biblioteka STL, testuje poprawność programów dla różnych danych, posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Efekty:

- umie uruchomić potrzebne oprogramowanie,
- umie napisać program z wykorzystaniem funkcji rekurencyjnej, równaniami rekurencyjnymi,
- zna struktury danych: tablica prefiksowa,
- zna porządek częściowy i liniowy,
- zna wyszukiwanie binarne

Formy i metody pracy: praca samodzielna w formie zawodów, sparingu

Zadania do wykonania na zajęciach	Treści programowe
1. Dobro i zło	M.3, P.2.13, P.2.16, A.3.2, A.4
2. Bursztyn	M.3, P.2.16, P.2.19, A.3.2, A.4
3. Na rzymskie	M.3, P.2.16, P.2.20, A.3.2, A.4

Podpowiedzi do rozwiązań

Zadanie 1. Dobro i zło. Proste wyszukiwanie binarne po wyniku. Utrudnieniem jest konieczność skorzystania z biblioteczki.

Zadanie 2. Bursztyn. Zadanie należy rozwiązać w oparciu o programowanie dynamiczne. Najłatwiej wykorzystać sumy prefiksowe w tablicy dwuwymiarowej.

Zadanie 3. Na rzymskie. Wystarczające będzie użycie algorytmu zachłannego. Przygotujmy sobie dwie tablice ze wszystkimi możliwymi wartościami „cyfr” oraz cyframi w zapisie rzymskim, a następnie wypisujmy każdą „cyfrę” odpowiednią liczbę razy od począwszy tej o największej wartości.

ZADANIA I ROZWIĄZANIA:

Zadanie 1. Dobro i zło

Limit pamięci: 64MB Limit czasu: 0.5s

Walka dobra ze złem w Vilolandzie trwa w najlepsze. Niedługo rozpęta się bitwa, która zadecyduje o wszystkim, tylko... no właśnie... gdzie mogłaby się odbyć?

Viloland, jak wiadomo, jest światem w kształcie prostokąta podzielonego na 1 x n kwadratowych pól. Każde pole jest kontrolowane albo przez dobro, albo przez zło, przy czym wiadomo, że pole numer 1 jest kontrolowane przez dobro, a pole numer n przez zło. To, przez kogo jest kontrolowane dane pole, może się dowolnie przeplatać i każde ze „środkowych" $n - 2$ pól nie jest w żaden sposób zależne od innych.

Wielka bitwa musi odbyć się na dwóch sąsiednich polach, z których lewe (to o mniejszym indeksie) jest kontrolowane przez dobro, a prawe przez zło. Póki co przywódcy obu stron tylko się kłócą, ale w końcu bitwa musi się odbyć. Pomóż znaleźć odpowiednie miejsce!

Komunikacja

Twój program powinien używać biblioteki, która pozwala na zadawanie pytań o to, kto kontroluje poszczególne pola, oraz na udzielenie ostatecznej odpowiedzi. Aby użyć biblioteki, należy wpisać na początku programu:

```
#include "doblib.h"
```

Biblioteka udostępnia następujące funkcje:

- `long long inicjuj();`

  Inicjalizuje bibliotekę. Należy wywołać ją tylko raz, na początku programu, przed użyciem pozostałych funkcji. Zwraca liczbę n, oznaczającą liczbę pól w Vilolandzie.

- `int sily(long long x);`

  Dla każdego x spełniającego warunek $1 \leq x \leq n$ zwraca 1, jeśli pole numer x jest kontrolowane przez siły dobra, 2, jeśli przez siły zła. W szczególności `sily(1)` zwróci 1, `sily(n)` zwróci 2. **Uwaga! Możesz użyć tej funkcji co najwyżej 100 razy!**

- `void odpowiedz(long long x);`

  Udziela odpowiedzi, że na polu numer x panują siły dobra, zaś na polu numer $x + 1$ - siły zła. Wywołanie tej funkcji zakończy działanie twojego programu.

Twój program **nie może** czytać żadnych danych (ani ze standardowego wejścia, ani z plików). **Nie może** również nic wypisywać do plików ani na standardowe wyjście. Może pisać na standardowe wyjście diagnostyczne (`stderr`) - pamiętaj jednak, że zużywa to cenny czas.

Przykładowy przebieg programu

Załóżmy, że $n = 7$, a rozkład sił na kolejnych polach wygląda tak:

1121222

gdzie podobnie jak wyżej 1 oznacza kontrolę sił dobra, a 2 - sił zła. Poprawne odpowiedzi to $x = 2$ oraz $x = 4$. Interakcja programu z biblioteką może wyglądać następująco:

Wywołanie	Zwrócona wartość
`inicjuj()`	-
`sily(2)`	1
`sily(3)`	2
`odpowiedz(2)`	-

Ocenianie

Podzadanie	Ograniczenia	Punkty
1	$n \leq 10^2$	20
2	$n \leq 10^{18}$	80

Eksperymenty

W zakładce Pliki znajduje się archiwum `dob_dlazaw.zip` zawierające:

- Przykładową bibliotekę `doblib.h`, która pozwoli Ci przetestować poprawność formalną rozwiązania. Aby jej użyć, umieść ją w tym samym folderze co swoje rozwiązanie, a następnie skompiluj je. Tak otrzymany program wczytuje z wejścia liczbę *n*, a następnie ciąg *n* znaków równych '1' lub '2', w zależności od tego czy dane pole kontrolowane jest przez siły dobra, czy przez siły zła. Pamiętaj jednak, że biblioteka ta różni się od tej, na której zostanie ostatecznie ocenione twoje rozwiązanie.
- Przykładowe, błędne rozwiązanie `dob.cpp`, ilustrujące poprawną komunikację z biblioteką.
- Test przykładowy `dob0.in` w opisanym wyżej formacie przykładowej biblioteki.

Zadanie 2. Bursztyn

Dostępna pamięć:32 MB

Czy wiedzieliście, że w województwie lubelskim są ogromne złoża bursztynu, porównywalne z tymi na wybrzeżu Bałtyku? Wójt jednej z lubelskich gmin postanowił na własną rękę eksploatować złoża znajdujące się na zarządzanym przez niego terenie. W tym celu zabezpieczył już w przyszłorocznym budżecie pewną kwotę na zakup terenów bursztynonośnych. Ze względu na zakupiony sprzęt teren ten musi mieć kształt kwadratu o określonym wymiarze *k*. Wójt sporządził już mapę terenu z informacjami o ilości bursztynu na poszczególnych działkach. Teraz tylko chciałby wybrać taki kwadrat, którego eksploracja przyniesie mu największy zysk.

Wejście

W pierwszym wierszu standardowego wejścia zapisano trzy liczby całkowite dodatnie, oddzielone pojedynczym odstępem: *n* , *m* i *k*, gdzie ($1 \leq n, m, k \leq 1000$). W kolejnych *n* wierszach zapisano po *m* liczb całkowitych nieujemnych, które odpowiadają ilości bursztynu na poszczególnych działkach (wielkości te nie przekraczają miliona).

Wyjście

W pierwszym wierszu standardowego wyjścia zapisz największą sumaryczną ilość bursztynu kwadratowego obszaru.

Przykłady

Wejście	Wejście	Wejście
5 6 2 1 2 3 4 5 6 5 4 3 2 1 6 3 4 5 6 7 1 4 5 6 7 8 1 8 9 1 2 3 1	4 5 3 1 2 3 4 5 5 4 3 2 1 3 4 5 6 7 4 5 6 7 8	4 4 2 1 2 3 4 5 4 3 2 3 4 5 6 4 5 6 7
Wyjście 28	Wyjście 45	Wyjście 24

Zadanie 3. Na rzymskie

Dostępna pamięć: 32MB

Wczytaj liczbę zapisaną w systemie dziesiętnym i wypisz jej wartość zapisaną cyframi rzymskimi. Wartości cyfr pojawiających się w testach przedstawione są w tabelce poniżej.

I	1
IV	4
V	5
IX	9
X	10
L	50
C	100
D	500
M	1000

Wejście

Liczba arabska n ($1 \leq n \leq 3000$).

Wyjście

Pierwszy i jedyny wiersz wyjścia powinien wypisać jedno słowo – wartość n w zapisie rzymskim.

Przykład

Wejście	Wyjście
55	LV