16 listopada 2015

Zadanie 5

Przeczytaj rozdziały o wskaźnikach funkcyjnych, lambdach i wzorcach funkcji (rozdziały 11.12—11.14).

W poniższym kodzie szablon funkcji `part` deklaruje jako trzeci argument (`FUN`) *cokolwiek* co da się wywołać (wskaźnik funkcyjny, lambda) z argumentem typu `T` i zwraca `bool` (takie funkcje nazywamy *predykatami*).

Uzupełnij kod programu, tak, aby dał się skompilować i wykonać.

Funkcja (szablon) `printTab` ma za zadanie wydrukować w ładnej formie przekazaną tablicę (elementy w jednym wierszu, oddzielone spacjami).

Funkcja (szablon) `part` ma za zadanie tak poprzestawiać elementy przekazanej tablicy `arr`, aby wszystkie elementy, dla których predykat `FUN` jest spełniony (tzn. `FUN` wywołany z wartością tego elementu jako argumentem zwraca `true`) znalazły się na lewo od wszystkich elementów, dla których ten predykat nie jest spełniony. Jako argumentu odpowiadającego parametrowi `FUN` można użyć wskaźnika do funkcji typu `T→bool` (jak w linii 23) lub lambdy o takiej sygnaturze. W wynikowej tablicy względna kolejność elementów w ramach tych, które predykat spełniają i tych, które go nie spełniają, jest dowolna. Funkcja zwraca indeks pierwszego elementu, który *nie* spełnia predykatu; zauważ, że jest to jednocześnie liczba elementów, które predykat *spełniają* (być może 0 lub `size`).

```
1   #include <iostream>
2   #include <cstdlib>
3   #include <functional>
4   using namespace std;
5
6   template <typename T, typename FUN>
7   size_t part(T* arr, size_t size, FUN f) {
8       // ...
9   }
10
11  template <typename T>
12  void printTab(const T* t, size_t size) {
13      // ...
14  }
15
16  bool isEven(int e) { return e%2 == 0; }
17
18  int main() {
19
```

```
20          size_t ind = 0;
21
22          int a1[] = {1,2,3,4,5,6};
23          ind = part(a1,6,isEven);
24          cout << "ind = " << ind << " ";
25          printTab(a1,6);
26
27          int a2[] = {1,2,3,4,5,6};
28          // lambda jako argument: predykat sprawdzajacy,
29          // czy podana liczba jest nieparzysta
30          ind = part( /* ... */ );
31          cout << "ind = " << ind << " ";
32          printTab(a2,6);
33
34          double a3[] = {-1.5,2.5,3.5,6.5,4.5,0};
35          double mn =2.0;
36          double mx =5.0;
37          // lambda jako argument: predykat sprawdzajacy, czy
38          // podana liczba miesci sie w przedziale [mn.mx]
39          ind = part( /* ... */ );
40          cout << "ind = " << ind << " ";
41          printTab(a3,6);
42
43          constexpr size_t DIM = 500000;
44          int* a4 = new int[DIM];
45          srand(unsigned(time(0)));
46          for (size_t i = 0; i < DIM; ++i) a4[i] = rand()%21+1;
47          // lambda jako argument: predykat sprawdzajacy,
48          // czy podana liczba dzieli sie przez 7
49          ind = part( /* ... */ );
50          cout << "ind = " << ind << endl;
51          delete [] a4;
52      }
```

Fragment w liniach 43-46 służy do wygenerowania wartości do zainicjowania tablicy `a4` (wszystkie wartości będą pochodzić z przedziału $[1, 21]$).

Operację rozdzielania elementów należy przeprowadzić w *jednej* pojedynczej pętli (bez pętli zagnieżdżonych), inaczej wywołanie z linii 49, dla tablicy o wymiarze pół miliona, trwałoby zbyt długo; przy poprawnej implementacji wykonanie powinno być praktycznie natychmiastowe.

Przykładowy wynik programu mógłby wyglądać tak:

```
ind = 3 [ 2 4 6 1 5 3 ]
ind = 3 [ 1 3 5 4 2 6 ]
ind = 3 [ 2.5 3.5 4.5 6.5 -1.5 0 ]
ind = 71461
```

Termin: do 30 listopada (włącznie)

Rozwiązania, w postaci ***jednego*** *pliku źródłowego zawierającego treść programu, proszę wrzucać w systemie EDU do katalogu „Foldery zadań / Zadanie_XX", gdzie 'XX' jest numerem zadania.*

Nazwą pliku powinno być nazwisko z dużej litery (bez polskich znaków); rozszerzeniem musi być '`.cpp`*', czyli np.* `Malinowska.cpp`*.*