Podstawy Programowania
Semestr letni 2022/23
Materiały z laboratorium i zadania domowe

Przemysław Olbratowski

3 marca 2023

Slajdy z wykładu są dostępne w serwisie UBI. Informacje organizacyjne oraz formularz do uploadu prac domowych znajdują się na stronie `info.wsisiz.edu.pl/~olbratow`. Przy zadaniach domowych w nawiasach są podane terminy sprawdzeń.

14.2 Zadania domowe z działu Algorytmy (28 czerwca, 5, 12 lipca)

14.2.1 Combination: Losowa kombinacja

Napisz program `combination`, która czyta ze standardowego wejścia nieujemne liczby całkowite do napotkania końca pliku i wypisuje na standardowe wyjście w losowej kolejności tyle zer, jedynek i tak dalej, ile wynoszą kolejne wczytane liczby. Program zawiera po jednym wywołaniu funkcji `std::accumulate`, `std::fill_n` oraz `std::random_shuffle` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime`, `algorithm`, `iostream`, `numeric` i `vector`.

Przykładowe wykonanie

```
 In: 3 0 5 2
Out: 2 3 2 0 2 3 2 0 0 2
```

14.2.2 Duplicates: Duplikaty - indywidualnie

Napisz program `duplicates`, który czyta ze standardowego wejścia liczby całkowite do napotkania końca pliku i wypisuje na standardowe wyjście w kolejności rosnącej wszystkie liczby powtarzające się, każdą tylko raz. Program zawiera po jednym wywołaniu funkcji `std::adjacent_find`, `std::find_if` oraz `std::sort` i nie używa innych algorytmów bibliteki standardowej. Program załącza tylko pliki nagłówkowe `algorithm`, `iostream` i `vector`.

Przykładowe wykonanie

```
 In: 13 7 2 13 5 2 1 13
Out: 2 13
```

14.2.3 Iota: Uogólnienie algorytmu iota

Napisz funkcję `iota`, która przyjmuje modyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych, a także wartość początkową oraz krok i wpisuje do kolejnych komórek wycinka liczby z zadanym krokiem począwszy od podanej wartości początkowej. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja zawiera jedno wywołanie algorytmu `std::for_each` i nie używa innych algorytmów biblioteki standardowej. Funkcja korzysta tylko z plików nagłówkowych `algorithm` i `vector`.

Przykładowy program

```
int main() {
    std::vector<int> vector(10);
    iota(vector.begin(), vector.end(), 7, 2);
    std::for_each(vector.begin(), vector.end(),
                  [](int element){std::cout << element << " "; });
    std::cout << std::endl; }
```

Wykonanie

```
Out: 7 9 11 13 15 17 19 21 23 25
```

14.2.4 Median: Mediana

Medianę z kilku liczb rzeczywistych obliczamy następująco. Zapisujemy liczby w kolejności niemalejącej. Jeżeli jest ich nieparzysta liczba, to mediana jest środkową z nich. W przeciwnym razie mediana jest średnią arytmetyczną dwóch środkowych liczb. Napisz program `median`, który czyta za standardowego wejścia liczby rzeczywiste do napotkania końca pliku i wypisuje na standardowe wyjście ich medianę. Program zawiera jedno wywołanie funkcji `std::sort` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `algorithm`, `iostream` i `vector`.

Przykładowe wykonanie

```
 In: 10.5 -12.3 5.7 9.9 -1 15
Out: 7.8
```

14.2.5 Memory: Gra karciana memory

W prostej jednoosobowej wersji gry memory rekwizytami są dwie karty z literą A, dwie z literą B i tak dalej. Karty tasujemy i układamy w jednym rzędzie koszulkami do góry. Odkrywamy losowo dwie karty. Jeżeli są na nich różne litery, to zakrywamy je z powrotem, a w przeciwnym razie zabieramy. Celem gry jest zabranie wszystkich kart w jak najmniejszej liczbie prób. Napisz program `memory` symulujący taką grę. Program wyświetla aktualne ułożenie kart w następujący sposób:

```
Out:  1  B     C     6
```

Liczby to kolejne numery kart licząc od lewej, litery to odkryte karty, a puste miejsca to już zabrane karty. Program przyjmuje jako argument wywołania liczbę liter, tasuje karty i wyświetla początkowe ułożenie ze wszystkimi kartami zakrytymi. Następnie wczytuje numery dwóch kart i wyświetla ułożenie z tymi kartami odkrytymi. Po trzech sekundach drukuje tyle pustych linii, aby dotychczasowy wydruk zniknął z ekranu. Następnie wyświetla kolejne ułożenie odpowiednio zakrywając lub usuwając ostatnio odkryte karty i tak dalej. Po zabraniu wszystkich kart program automatycznie kończy działanie.

Przykładowe wykonanie

```
  Linux: ./memory 2
Windows: memory.exe 2

Out: 1  2  3  4
 In: 1 2
Out: A  B  3  4

Out: 1  2  3  4
 In: 3 4
Out: 1  2  B  A

Out: 1  2  3  4
 In: 2 3
Out: 1  B  B  4

Out: 1        4
 In: 1 4
Out: A        A
```

14.2.6 Mode: Najczęściej występująca liczba

Napisz program `mode`, który czyta ze standardowego wejścia wartości całkowite do napotkania końca pliku i wypisuje na standardowe wyjście wartość najczęściej się powtarzającą oraz liczbę jej wystąpień. Jeżeli takich wartości jest kilka, program wypisuje najmniejszą z nich. Program zawiera jedno wywołanie funkcji `std::sort` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `algorithm`, `iostream` i `vector`.

Przykładowe wykonanie

```
 In: 2 1 7 7 5 2 0 2 7 1
Out: 2 3
```

14.2.7 Neighbors: Najbliżsi sąsiedzi

Napisz program `neighbors`, który czyta ze standardowego wejścia liczby rzeczywiste do napotkania końca pliku i wypisuje na standardowe wyjście w kolejności niemalejącej te dwie spośród nich, które się najmniej różnią. Program zawiera jedno wywołanie funkcji `std::sort` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `cmath`, `algorithm`, `iostream` i `vector`.

Przykładowe wykonanie

```
 In: 10 -1.2 2.3 5.5 -5.5 7.1 0.4
Out: 5.5 7.1
```

14.2.8 Nth Element: Element n-ty co do wartości

Napisz program `nth_element`, który przyjmuje jako argument wywołania nieujemną liczbę całkowitą n, czyta ze standardowego wejścia liczby rzeczywiste do napotkania końca pliku i wypisuje na standardowe wyjście liczbę n-tą co do wartości. Program zawiera jedno wywołanie funkcji `std::nth_element` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `cstdlib`, `algorithm`, `iostream` i `vector`.

Przykładowe wykonanie

```
  Linux: ./selection 5
Windows: selection.exe 5
 In: 12.3 0.7 -0.1 8.3 5.7 0.7 8.3
Out: 8.3
```

14.2.9 Parity: Sortowanie według parzystości

Napisz program `parity`, który czyta ze standardowego wejścia liczby całkowite do napotkania końca pliku i wypisuje na standardowe wyjście najpierw wszystkie parzyste w kolejności niemalejącej, a potem wszystkie nieparzyste w takiej samej kolejności. Program zawiera jedno wywołanie funkcji `std::sort` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `algorithm` i `vector`.

Przykładowe wykonanie

```
 In: 1 12 5 17 3 10 7 8
Out: 8 10 12 1 3 5 7 17
```

14.2.10 Partial Sum: Sumy częściowe

Napisz funkcję `partial_sum`, która przyjmuje modyfikujący iterator początkowy i końcowy wycinka wektora liczb całkowitych i do każdego elementu tego wycinka dodaje sumę wszystkich go poprzedzających. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja zawiera jedno wywołanie algorytmu `std::for_each` i nie używa innych algorytmów biblioteki standardowej. Funkcja korzysta tylko z plików nagłówkowych `algorithm` i `vector`.

Przykładowy program

```
int main() {
    std::vector<double> vector {1.2, -0.5, 0.3, 2.5, -1};
    partial_sum(vector.begin(), vector.end());
    std::for_each(vector.begin(), vector.end(),
        [](double element){std::cout << element << " "; });
    std::cout << std::endl; }
```

Wykonanie

```
Out: 1.2 0.7 1 3.5 2.5
```

14.2.11 Permutation: Losowa permutacja - grupowo

Napisz program `permutation`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą i wypisuje na standardowe wyjście losową permutację mniejszych od niej nieujemnych liczb całkowitych. Program zawiera po jednym wywołaniu funkcji `std::iota` oraz `std::random_shuffle` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime`, `algorithm`, `iostream`, `numeric` i `vector`.

Przykładowe wykonanie

```
 In: 5
Out: 2 0 4 1 3
```

14.2.12 Permutations: Wszystkie permutacje

Napisz program `permutations`, który wczytuje ze standardowego wejścia nieujemną liczbę całkowitą i wypisuje na standardowe wyjście wszystkie permutacje mniejszych od niej nieujemnych liczb całkowitych. Program zawiera po jednym wywołaniu funkcji `std::iota` oraz `std::next_permutation` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime`, `algorithm`, `iostream`, `numeric` i `vector`.

Przykładowe wykonanie

```
 In: 3
Out: 0 1 2
Out: 0 2 1
Out: 1 0 2
Out: 1 2 0
Out: 2 0 1
Out: 2 1 0
```

14.2.13 Puzzle: Układanka piętnastka

Układanka *Piętnastka* składa się z planszy 4×4, na której znajduje się 15 kwadratowych klocków ponumerowanych od 1 do 15 oraz jedno puste miejsce. Celem gry jest uporządkowanie wymieszanych klocków poprzez przesuwanie ich po jednym na puste miejsce, przy czym przesunąć można tylko klocek sąsiadujący z tym miejscem. Napisz program `puzzle` symulujący taką grę. Po uruchomieniu program rozmieszcza klocki losowo, wyświetla początkowe ułożenie, a następnie wczytuje z klawiatury numer klocka. Jeśli to możliwe, przesuwa go na puste miejsce, drukuje planszę po przesunięciu, i tak dalej. Po uporządkowaniu klocków program automatycznie kończy wykonanie.

Końcówka przykładowej rozgrywki

```
Out: 01 02 03 04
Out: 05 06 07 08
Out: 09 10 11 12
Out: 13 14    15

 In: 15

Out: 01 02 03 04
Out: 05 06 07 08
Out: 09 10 11 12
Out: 13 14 15
```

14.2.14 Tictactoe: Kółko i krzyżyk

Grę kółko i krzyżyk na klasycznej planszy 3×3 zaczynają zawsze krzyżyki. Napisz program `tictactoe`, który przyjmuje jako argument wywołania znak użytkownika, `X` lub `O`, po czym wyświetla pustą planszę. Jeżeli wypada ruch użytkownika, program wczytuje współrzędne pola, na którym użytkownik stawia swój znak, na przykład `b3`. Jeżeli wypada ruch programu, wypisuje współrzędne pola, na którym sam stawia swój znak. Potem wyświetla odpowiednio zaktualizowaną planszę i tak dalej. Program sam wykrywa zakończenie gry i wypisuje informację o wyniku. Program nigdy nie przegrywa i wygrywa zawsze, kiedy to możliwe. Program załącza tylko pliki nagłówkowe `cstdlib`, `algorithm`, `iostream` i `vector`.

Początek przykładowego wykonania

```
  Linux: ./tictactoe X
Windows: tictactoe.exe X

Out:  abc
Out: 1
Out: 2
Out: 3

 In: a1

Out:  abc
Out: 1X
Out: 2
Out: 3

Out: c2

Out:  abc
Out: 1X
Out: 2  O
Out: 3
```

14.2.15 Triangles: Liczba trójkątów

Napisz program `triangles`, który czyta ze standardowego wejścia długości odcinków do napotkania końca pliku i wypisuje na standardowe wyjście liczbę trójkątów, które można z nich zbudować. Każdego odcinka można użyć w jednym trójkącie tylko raz, a dwa odcinki o jednakowych długościach uważamy za różne. Program zawiera jedno wywołanie funkcji `std::sort` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `algorithm`, `iostream` i `vector`.

Przykładowe wykonanie

```
 In: 2.5 1 3.7 3
Out: 3
```

14.2.16 Variation: Losowa wariacja

Napisz program `variation`, który wczytuje ze standardowego wejścia nieujemne liczby całkowite k oraz n i drukuje na standardowe wyjście k niepowtarzających się losowych, nieujemnych liczb całkowitych mniejszych od n. Program zawiera po jednym wywołaniu funkcji `std::iota` oraz `std::random_shuffle` i nie używa innych algorytmów biblioteki standardowej. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime`, `algorithm`, `iostream`, `numeric` i `vector`.