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.

15.2 Zadania domowe z działu Pojemniki

15.2.1 Change: Problem wydawania reszty

Napisz program `change` odliczający zadaną kwotę z posiadanej gotówki w możliwie najmniejszej liczbie banknotów i monet. Kwota zawiera tylko pełne złote zaś w portfelu nie ma groszy. Program przyjmuje jako argument wywołania kwotę, po czym czyta ze standardowego wejścia nominały posiadanych banknotów i monet do napotkania końca pliku. Następnie wypisuje na standardowe wyjście odliczone nominały. Jeżeli żądanej kwoty nie da się odliczyć z posiadanej gotówki, program nic nie wypisuje. Program załącza tylko pliki nagłówkowe `cstdlib`, `iostream` i `set`.

Przykładowe wykonanie

```
  Linux: ./change 67
Windows: change.exe 67
 In: 50 20 20 20 5 5 1 1
Out: 20 20 20 5 1 1
```

Wskazówka Zaimplpementuj następujący algorytm rekurencyjny. Jeżeli żądana kwota jest równa któremuś z posiadanych nominałów, to nominał ten wyjmujemy z portfela i kończymy procedurę sukcesem. Jeżeli w portfelu nie ma nominałów mniejszych od danej kwoty, to kończymy procedurę porażką. W przeciwnym razie wyjmujemy z portfela największy nominał mniejszy od danej kwoty, samą kwotę pomniejszamy o ten nominał i rozpoczynamy procedurę od początku. Jeżeli odliczenie pomniejszonej kwoty z pozostałych w portfelu nominałów powiedzie się, to kończymy procedurę sukcesem. W przeciwnym wypadku ostatnio wyjęty nominał chowamy z powrotem do portfela i odpowiednie czynności powtarzamy dla kolejnego, mniejszego nominału.

15.2.2 Decays: Rozpady promieniotwórcze

Plik `decays.txt` wylicza wszystkie możliwe rozpady wszystkich znanych nuklidów promieniotwórczych. Każda linia zawiera informacje o rozpadach jednego nuklidu. Przykładowo, linia

```
179Hg 175Pt 178Pt 179Au
```

oznacza, że ^{179}Hg może się rozpaść do ^{175}Pt, ^{178}Pt lub ^{179}Au. Napisz program `decays`, który wczytuje ze standardowego wejścia dowolną listę nuklidów i wypisuje na standardowe wyjście wszystkie nuklidy, które mogą z nich powstać w wyniku wielokrotnych rozpadów promieniotwórczych, przy czym każdy nuklid wypisuje tylko raz. Program załącza tylko pliki nagłówkowe `fstream`, `iostream`, `map`, `set`, `sstream` i `string`.

Przykładowe wykonanie

```
 In: 235U 238U
Out: 206Hg 206Pb 206Tl 207Pb 207Tl 209Bi 209Pb 210Bi 210Pb 210Po 210Tl 211Bi 211Pb
     211Po 214Bi 214Pb 214Po 215At 215Bi 215Po 218At 218Po 218Rn 219At 219Rn 222Rn
     223Fr 223Ra 226Ra 227Ac 227Th 230Th 231Pa 231Th 234Pa 234Th 234U 235U 238U
```

15.2.3 Intersection: Część wspólna zbiorów

Napisz funkcję `intersection`, która przyjmuje stałe referencje dwóch zbiorów liczb całkowitych i zwraca ich część wspólną. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja wywołuje na zbiorach tylko metody `cbegin`, `cend` oraz `insert` i korzysta tylko z pliku nagłówkowego `set`.

Przykładowy program

```
int main () {
    std::set<int> set1 {2, 3, 23, 0, 1, 2, -7}, set2 {15, 19, 23, -7, 23, 1, -19};
    std::set<int> result = intersection(set1, set2);
```

```
    for (int element: result) {
        std::cout << element << " "; }
    std::cout << std::endl; }
```

Przykładowe wykonanie

```
Out: -7 1 23
```

15.2.4 Makao: Gra karciana makao

Jednoosobowa wersja karcianej gry makao mogłaby wyglądać następująco. Tasujemy całą talię, 6 kart bierzemy do ręki, a resztę kładziemy na kupce awersami do góry. Karty z ręki odkładamy po jednej na kupkę do karcianego koloru lub wartości. Jeżeli nie mamy pasującej karty, dobieramy jedną ze spodu kupki i tak dalej. Celem gry jest pozbycie się wszystkich kart z ręki w możliwie najmniejszej liczbie prób. Napisz program `makao` symulujący taką grę. Po uruchomieniu program tasuje i rozdaje karty. Następnie wypisuje na standardowe wyjście kartę z wierzchołka kupki, znak `|`, oraz karty na ręce w kolejności trefl, karo, kier, pik, zaś w każdym kolorze od dwójki do asa. Jeżeli na ręce nie ma pasującej karty, program sam dobiera jedną ze spodu kupki i wypisuje ją po znaku `<-`. Czynność tę powtarza aż na ręce znajdzie się pasująca karta. Jeżeli na ręce jest przynajmniej jedna pasująca karta, program wypisuje znak `->` i wczytuje ze standardowego wejścia kartę do odłożenia na kupkę. Czynności te powtarza aż do wyczerpania kart na ręce, po czym kończy działanie.

Przykładowa rozgrywka

```
Out: JH | 7C AD 2S 8S XS AS <- KD
Out: JH | 7C KD AD 2S 8S XS AS <- KS
Out: JH | 7C KD AD 2S 8S XS KS AS <- 8C
Out: JH | 7C 8C KD AD 2S 8S XS KS AS <- 5S
Out: JH | 7C 8C KD AD 2S 5S 8S XS KS AS <- JS
Out: JH | 7C 8C KD AD 2S 5S 8S XS JS KS AS -> In: JS
Out: JS | 7C 8C KD AD 2S 5S 8S XS KS AS -> In: AS
Out: AS | 7C 8C KD AD 2S 5S 8S XS KS -> In: AD
Out: AD | 7C 8C KD 2S 5S 8S XS KS -> In: KD
Out: KD | 7C 8C 2S 5S 8S XS KS -> In: KS
Out: KS | 7C 8C 2S 5S 8S XS -> In: XS
Out: XS | 7C 8C 2S 5S 8S -> In: 5S
Out: 5S | 7C 8C 2S 8S -> In: 2S
Out: 2S | 7C 8C 8S -> In: 8S
Out: 8S | 7C 8C -> In: 8C
Out: 8C | 7C -> In: 7C
Out: 7C |
```

15.2.5 Mastermind: Master Mind

Mastermind to gra polegająca na odgadywaniu ułożenia k pionków numerowanych od 1 do n. Pionki umieszcza się w rzędzie od lewej do prawej. Zależnie od umowy, pionek o tym samym numerze może się powtarzać lub nie. Pierwszy gracz układa w ukryciu pionki, na przykład `6 3 1 2`. Aby odgadnąć to ułożenie, drugi gracz pokazuje pierwszemu dowolne ułożenie pionków, na przykład `2 6 1 6`. W odpowiedzi pierwszy informuje go, że aby z tego ułożenia uzyskać ukryte, powinien a pionków pozostawić na dotychczasowym miejscu, b pionków przestawić, a resztę wymienić na inne. W przedstawionym przykładzie należy pozostawić jedynkę na trzeciej pozycji, dwójkę i jedną szóstkę przestawić, a jedną szóstkę wymienić na inny numer, więc $a = 1$ i $b = 2$. Pierwszy gracz przekazuje drugiemu jedynie wartości a i b, nie ujawniając, które pionki należy pozostawić lub przestawić. Znając a oraz b zgadujący przedstawia kolejne próbne ułożenie i tak dalej, aż do odgadnięcia ukrytego ułożenia. Napisz program `mastermind` odgadujący ukryte ułożenie pionków. Przed przystąpieniem do gry użytkownik zapisuje na kartce ukryte ułożenie. Program przyjmuje jako argumenty wywołania liczby k i n oraz 1 lub 0 dla gry z powtórzeniami

lub bez. Po uruchomieniu wypisuje na standardowe wyjście próbne ułożenie i wczytuje ze standardowego wejścia odpowiadające mu wartości a oraz b. Czynności te powtarza aż wartości te będą świadczyć o odgadnięciu ukrytego ułożenia. Poniższy wydruk przedstawia przykładowe wykonanie programu przy ukrytym ułożeniu `6 3 1 2`. Program załącza tylko pliki nagłówkowe `cstdlib`, `iostream`, `set`, `utility` i `vector`.

Przykładowa rozgrywka

```
  Linux: ./mastermind 4 6 1
Windows: mastermind.exe 4 6 1
Out: 1 1 1 1  In: 1 0
Out: 1 2 2 2  In: 1 1
Out: 3 1 2 3  In: 0 3
Out: 4 2 3 1  In: 0 3
Out: 5 3 1 2  In: 3 0
Out: 6 3 1 2  In: 4 0
```

15.2.6 People: Sortowanie listy osób

Pewien plik tekstowy zawiera lata urodzenia i nazwiska osób zapisane jak poniżej. Napisz program `people`, który wczytuje ten plik ze standardowego wejścia i wypisuje na standardowe wyjście tę samą listę osób posortowaną według roku urodzenia, zaś w ramach jednego roku według kolejności alfabetycznej nazwisk. Program załącza tylko pliki nagłówkowe `iostream`, `set`, `string` i `utility`.

Przykładowe wykonanie

```
 In: 1879 Trotsky
 In: 1928 Warhol
 In: 1879 Einstein
 In: 1810 Chopin
 In: 1867 Curie
 In: 1928 Guevara
 In: 1879 Hahn
 In: 1867 Toscanini
 In: 1928 Nash
 In: 1810 Schumann

Out: 1810 Chopin
Out: 1810 Schumann
Out: 1867 Curie
Out: 1867 Toscanini
Out: 1879 Einstein
Out: 1879 Hahn
Out: 1879 Trotsky
Out: 1928 Guevara
Out: 1928 Nash
Out: 1928 Warhol
```

15.2.7 Permutation: Wykrywanie permutacji

Napisz program `permutation`, który czyta ze standardowego wejścia liczby całkowite do napotkania końca pliku i wypisuje na standardowe wyjście `true` jeżeli stanowią one permutację kolejnych liczb albo `false` w przeciwnym razie. Program załącza tylko pliki nagłówkowe `iostream` i `set`.

Przykładowe wykonanie

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

15.2.8 Phrase: Najczęstsza fraza

Pewien plik tekstowy zawiera tylko jedną linię. Napisz program `phrase`, który przyjmuje jako argument wywołania dodatnią liczbę całkowitą n, wczytuje plik ze standardowego wejścia i wypisuje na standardowe wyjście wszystkie najczęściej występujące w nim sekwencje n znaków ze znakami białymi włącznie. Program wypisuje te sekwencje w kolejnych wierszach według ich kolejności alfabetycznej. W ostatnim wierszu wypisuje liczbę wystąpień każdej z nich. Jeżeli we wczytanym pliku nie występuje żadna sekwencja o długości n, program nic nie wypisuje. Program załącza tylko pliki nagłówkowe `cstdlib`, `iostream`, `map` i `string`. Program przetestuj na tekście *Pana Tadeusza* i *Hamleta*.

Przykładowe wykonanie

```
  Linux: ./phrase 15 < hamlet.txt
Windows: phrase.exe 15 < hamlet.txt
Out:  fathers death
Out:  your lordship
Out: 8
```

15.2.9 Roman: Liczby rzymskie

Napisz program `roman`, który wczytuje ze standardowego wejścia dodatnią liczbę całkowitą podaną w zapisie arabskim lub rzymskim i wypisuje na standardowe wyjście tę samą liczbę w drugim zapisie. Program załącza tylko pliki nagłówkowe `iostream`, `map`, `sstream` i `string`.

Przykładowe wykonanie

```
 In: MMXVII
Out: 2017
```

15.2.10 Scores: Punkty w grze

Napisz program `scores`, który czyta ze standardowego wejścia pary złożone z nazwy gracza oraz liczby zdobytych prze niego punktów. Po napotkaniu końca pliku program wypisuje na standardowe wyjście w kolejności alfabetycznej nazwy wszystkich graczy oraz całkowite liczby zdobytych przez nich punktów. Program załącza tylko pliki nagłówkowe `iostream`, `map` i `string`.

Przykładowe wykonanie

```
 In: potter  9
 In: vader   4
 In: gandalf 4
 In: vader   6
 In: gandalf 7
Out: gandalf 11
Out: potter 9
Out: vader 10
```

15.2.11 Sudoku: Sudoku

Napisz program `sudoku`, który wczytuje ze standardowego wejścia początkową planszę klasycznego sudoku $3^2 \times 3^2$ i wypisuje na standardowe wyjście jego rozwiązanie.

Przykładowe wykonanie

```
 In: 5  64 7
 In:  4 7  258
 In: 8 9 35 4
 In:  2 8 7  3
```

```
 In: 93   452
 In: 65 92   4
 In:   5 8  71
 In: 7 1 9 4 5
 In: 4 3 7 6

Out: 512648739
Out: 346719258
Out: 879235146
Out: 124857963
Out: 938164527
Out: 657923814
Out: 295486371
Out: 761392485
Out: 483571692
```

15.2.12 Units: Zamiana jednostek

Napisz program `units`, który wczytuje ze standardowego wejścia wartość pewnej wielkości fizycznej wraz z symbolem jednostki i wypisuje na standardowe wyjście tę samą wartość wyrażoną we wszystkich znanych mu jednostkach tej samej wielkości. Zaimplementuj sekundę `s`, minutę `m=60s` i godzinę `h=3600s` jako jednostki czasu oraz stopę `ft`, jard `yd=3ft` i milę angielską `mi=5280ft` jako jednostki odległości. Program załącza tylko pliki nagłówkowe `iostream`, `map`, `set`, `sstream` i `string`.

Przykładowe wykonanie

```
 In: 12yd
Out: 36ft 0.00681818mi 12yd
```

15.2.13 Words: Zliczanie słów

Napisz program `words`, który wczytuje ze standardowego wejścia dowolny plik tekstowy i wypisuje na standardowe wyjście wszystkie występujące w nim słowa w kolejności liczby wystąpień, a po każdym słowie podaje liczbę jego wystąpień. Słowa występujące w pliku tyle samo razy wypisuje w kolejności alfabetycznej. Program załącza tylko pliki nagłówkowe `iostream`, `map`, `string` i `utility`.

Przykładowe wykonanie

```
 In: jeden dwa trzy jeden dwa cztery dwa jeden dwa
Out: cztery 1
Out: trzy 1
Out: jeden 3
Out: dwa 4
```