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.

7.2 Zadania domowe z działu Pliki (10, 17, 24 maja)

7.2.1 ASCII: Tabela kodów ASCII

Napisz program `ascii`, który wypisuje na standardowe wyjście wszystkie znaki od wykrzyknika do litery zet wraz z ich kodami, sformatowane jak poniżej. Program załącza tylko plik nagłówkowy `iostream`.

Wykonanie

```
Out:  33 !
Out:  34 "
Out:  35 #
...
Out: 120 x
Out: 121 y
Out: 122 z
```

7.2.2 Caesar: Szyfr Cezara - grupowo

Kodowanie wiadomości tekstowej szyfrem Cezara z przesunięciem n polega na zastąpieniu każdej litery inną, znajdującą się w alfabecie n pozycji dalej. Przesunięcie n może być dodatnie lub ujemne, przy czym przyjmujemy, że za literą z wypada litera a, zaś przed literą a wypada litera z. Małe litery są kodowane jako małe, a duże jako duże. Inne znaki nie są szyfrowane. Napisz program `caesar`, który przyjmuje jako argumenty wywołania przesunięcie oraz nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego kodując ją szyfrem Cezara z zadanym przesunięciem. Program załącza tylko pliki nagłówkowe `cstdlib` i `fstream`.

Przykładowy plik wejściowy input.txt

```
Bwf Dbftbs!
Wfoj, wjej, wjdj!
```

Przykładowe wywołanie

```
  Linux: ./caesar -1 input.txt output.txt
Windows: caesar.exe -1 input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
Ave Caesar!
Veni, vidi, vici!
```

7.2.3 Camel: Kamelizacja wyrazów

Napisz program `camel`, który przyjmuje jako argumenty wywołania nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego zamieniając pierwszą literę każdego słowa na wielką, a następne litery na małe. Pozostałe znaki program przepisuje bez zmian. Program załącza tylko pliki nagłówkowe `cctype` i `fstream`.

Przykładowy plik wejściowy input.txt

```
AlA mA 15 LAT!
czy ala ma 17 lat?
```

Przykładowe wywołanie

```
  Linux: ./camel input.txt output.txt
Windows: camel.exe input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
Ala Ma 15 Lat!
Czy Ala Ma 17 Lat?
```

7.2.4 Cat: Łączenie plików tekstowych

Napisz program `cat`, który przyjmuje jako argumenty wywołania nazwy dowolnej liczby plików tekstowych i wypisuje zawartości kolejnych plików na standardowe wyjście nie wstawiając między nimi żadnych dodatkowych znaków. Program załącza tylko pliki nagłówkowe `fstream` i `iostream`.

Przykładowy plik wejściowy input1.txt

```
to samo
```

Przykładowy plik wejściowy input2.txt

```
lot do Londynu
```

Przykładowe wykonanie

```
  Linux: ./cat input1.txt input2.txt
Windows: cat.exe input1.txt input2.txt
Out: to samolot do Londynu
```

7.2.5 Char: Znak o zadanym kodzie

Napisz program `char`, który wczytuje ze standardoweego wejścia kod znaku i wypisuje ten znak na standardowe wyjście. Program załącza tylko plik nagłówkowy `iostream`.

Przykładowe wykonanie

```
 In: 82
Out: R
```

7.2.6 Departures: Godziny odjazdów

Pewien plik tekstowy zawiera godziny odjazdu pociągu zapisane w kolejności rosnącej jak poniżej. Napisz program `departures`, który przyjmuje nazwę takiego pliku jako argument wywołania. Po uruchomieniu wczytuje ze standardowego wejścia godzinę, na przykład `10:45` i wypisuje na standardowe wyjście godzinę odjazdu najbliższego pociągu. Potem znów wczytuje godzinę i tak dalej, do napotkania końca pliku. Jeżeli tego samego dnia nie ma już żadnego połączenia, program wypisuje godzinę odjazdu pierwszego pociągu następnego dnia. Program załącza tylko pliki nagłówkowe `fstream`, `iomanip`, `iostream` i `vector`.

Przykładowy plik wejściowy input.txt

```
6:50 8:27 10:30 12:05 13:48 16:07 18:00 21:30
```

Przykładowe wykonanie

```
  Linux: ./departures input.txt
Windows: departures.exe input.txt
 In: 10:45  Out: 12:05
 In: 13:48  Out: 13:48
 In: 22:03  Out: 6:50
```

Wskazówka Godzinę z minutami pamiętaj jako jedną liczbę całkowitą, której dwie ostatnie cyfry to minuty, a wcześniejsze to godzina, na przykład `8:05` to `805`.

7.2.7 Enumerate: Numerowanie linii - indywidualnie

Napisz program `enumerate`, który przyjmuje jako argumenty wywołania nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego dopisując na początku każdej linii jej kolejny numer poczynając od jednego. Program załącza tylko plik nagłówkowy `fstream`.

Przykładowy plik wejściowy input.txt

```
Ala ma kota.

To kot Ali.
```

Przykładowe wywołanie

```
  Linux: ./enumerate input.txt output.txt
Windows: enumerate.exe input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
1 Ala ma kota.
2
3 To kot Ali.
```

7.2.8 Is: Znaki białe, cyfry, małe i duże litery

Używając funkcji z pliku nagłówkowego `cctype` napisz program `is`, który wczytuje ze standardowego wejścia jeden znak bez opuszczania znaków białych i wypisuje na standardowe wyjście `digit`, `upper`, `lower` albo `other` jeśli jest on odpowiednio cyfrą, wielką literą, małą literą albo innym znakiem. Program załącza tylko pliki nagłówkowe `cctype` i `iostream`.

Przykładowe wykonanie

```
 In: a
Out: lower
```

7.2.9 Letters: Zliczanie liter

Napisz program `letters`, który czyta ze standardowego wejścia znaki do napotkania końca pliku, a następnie wypisuje na standardowe wyjście liczby wystąpień kolejnych liter pośród tych znaków. Program nie rozróżnia wielkich i małych liter oraz pomija wszelkie inne znaki. Program załącza tylko pliki nagłówkowe `cctype`, `iostream` i `vector`.

Przykładowe wykonanie

```
 In: Lorem ipsum dolor sit amet!
Out: 1 0 0 1 2 0 0 0 2 0 0 2 3 0 3 1 0 2 2 2 1 0 0 0 0 0
```

7.2.10 LF: Konwersja znaków końca linii

W systemie Linux koniec linii oznacza się pojedynczym znakiem *line feed*, czyli `\n`, zaś w systemie Windows używa się pary *carriage return - line feed*, czyli `\r\n`. Napisz program `lf`, który przyjmuje jako argumenty wywołania nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego zastępując pojedyncze znaki `\n` parami `\r\n` i na odwrót. Program załącza tylko plik nagłówkowy `fstream`.

Przykładowy plik wejściowy input.txt

```
Ala ma kota.[CR][LF]
To kot Ali.[LF]
```

Przykładowe wykonanie

```
  Linux: ./lf input.txt output.txt
Windows: lf.exe input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
Ala ma kota.[LF]
To kot Ali.[CR][LF]
```

7.2.11 Password: Losowe hasło

Napisz program `password`, który przyjmuje jako argument wywołania nieujemną liczbę całkowitą i wypisuje na standardowe wyjście takiej długości hasło złożone z losowych cyfr i małych liter. Program załącza tylko pliki nagłówkowe `cstdlib`, `ctime` i `iostream`.

Przykładowe wykonanie

```
  Linux: ./password 7
Windows: password.exe 7
Out: 5zy4hsu
```

7.2.12 Phones: Numery telefonów

Pewien plik tekstowy zawiera numery telefonów zapisane jak poniżej. Napisz program `phones`, który przyjmuje nazwę tego pliku jako argument wywołania, po uruchomieniu wczytuje ze standardowego wejścia sekwencję cyfr i wypisuje na standardowe wyjście wszystkie numery telefonów zawierające tę sekwencję. Następnie wczytuje kolejną liczbę i tak do napotkania końca pliku. Program załącza tylko pliki nagłówkowe `fstream`, `iostream` i `vector`.

Przykładowy plik wejściowy input.txt

```
72648120 5812419 4285725 3897124 159004621 8887133
```

Przykładowe wykonanie

```
  Linux: ./phones input.txt
Windows: phones.exe input.txt
 In: 12
Out: 72648120 5812419 3897124
 In: 124
Out: 5812419 3897124
```

7.2.13 Separate: Odstępy między wyrazami

Napisz program `separate`, który przyjmuje jako argumenty wywołania nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego zamieniając każdą sekwencję spacji i tabulatorów na pojedynczą spację. Program załącza tylko plik nagłówkowy `fstream`.

Przykładowy plik wejściowy input.txt

```
Ala       ma     kota.
To kot         Ali.
```

Przykładowe wywołanie

```
  Linux: ./separate input.txt output.txt
Windows: separate.exe input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
Ala ma kota.
To kot Ali.
```

7.2.14 Space: Zamiana tabulatorów na spacje

Napisz program `space`, który przyjmuje jako argumenty wywołania liczbę spacji w tabulatorze oraz nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego zastępując wszystkie tabulatory spacjami tak, aby w edytorze tekstu oba pliku wyglądały tak samo. Program załącza tylko pliki nagłówkowe `cstdlib` i `fstream`.

Przykładowy plik wejściowy input.txt

```
four[tab]three[tab]two[tab]one
```

Przykładowe wywołanie

```
  Linux: ./space 4 input.txt output.txt
Windows: space.exe 4 input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
four[space][space][space][space]three[space][space][space]two[space]one
```

7.2.15 Tabulate: Zamiana spacji na tabulatory

Napisz program `tabulate`, który przyjmuje jako argumenty wywołania liczbę spacji w tabulatorze oraz nazwy dwóch plików tekstowych i przepisuje zawartość pierwszego pliku do drugiego zastępując jak najwięcej spacji tabulatorami tak, aby w edytorze tekstu oba pliki wyglądały tak samo. Program załącza tylko pliki nagłówkowe `cstdlib` i `fstream`.

Przykładowy plik wyjściowy input.txt

```
four[space][space][space][space]three[space][space][space]two[space]one
```

Przykładowe wywołanie

```
  Linux: ./tabulate 4 input.txt output.txt
Windows: tabulate.exe 4 input.txt output.txt
```

Przykładowy plik wejściowy output.txt

```
four[tab]three[tab]two[tab]one
```

7.2.16 Upper: Zamiana małych liter na wielkie

Napisz funkcję `upper`, która przyjmuje jeden znak. Jeżeli jest on małą literą, zwraca odpowiadającą jej wielką literę. W przeciwnym razie zwraca niezmieniony znak. Funkcja powinna być przystosowana do użycia w przykładowym programie poniżej. Funkcja nie używa instrukcji warunkowej i nie korzysta z żadnych plików nagłówkowych.

Przykładowy program

```
int main() {
    std::cout << upper('k') << upper('R') << upper('5') << std::endl; }
```

Wykonanie

```
Out: KR5
```

7.2.17 XOR: Szyfrowanie bitową różnicą symetryczną

Znak można zaszyfrować biorąc bitową różnicę symetryczną jego kodu z zadaną liczbą całkowitą. Biorąc różnicę symetryczną uzyskanej wartości z tą samą liczbą odzyskuje się wyjściowy znak. Napisz program `xor`, który przyjmuje jako argumenty wywołania liczbę całkowitą oraz nazwy dwóch plików tekstowych. Jeżeli podana liczba jest dodatnia, program czyta z pierwszego pliku znaki, bierze ich różnicę symetryczną z podaną liczbą i uzyskane wartości zapisuje do drugiego pliku jako liczby. W przeciwnym razie program czyta z pierwszego pliku liczby, bierze ich różnicę symetryczną z liczbą przeciwną do podanej i uzyskane wartości zapisuje do drugiego pliku jako znaki. Program załącza tylko pliki nagłówkowe `cstdlib` i `fstream`.

Przykładowy plik wejściowy input.txt

```
October 2018
```

Przykładowe wywołanie

```
  Linux: ./xor 157 input.txt output.txt
Windows: xor.exe 157 input.txt output.txt
```

Przykładowy plik wyjściowy output.txt

```
210 254 233 242 255 248 239 189 175 173 172 165
```