Programowanie zaawansowane FM i IN 2017/2018

Zadania domowe – seria 2.

Termin: 22 maja 2018 r.

Rozwiązaniem każdego z zadań powinien być plik lub więcej plików zawierających kod w języku C++, mających standardowe rozszerzenia, np. `.cpp`, `.h`. Nazwa pliku powinna mieć postać `nazwisko-zadanie.roz`, np. `kowalski-calc.cpp`. W przypadku istnienia kilku plików o tych samych rozszerzeniach należy jeszcze dodać do nazwy ich numery: `nazwisko-zadanie-numer.roz`, np. `kowalski-calc-1.cpp`, `kowalski-calc-2.cpp`.

Zadanie 1. `intersection` – Część wspólna zbiorów liczb (1 pkt.)

Napisz program `intersection`, który wczytuje ze standardowego wejścia dwie linie liczb całkowitych, a następnie wypisuje na standardowe wyjście te liczby, które występują w obu liniach jednocześnie. Liczby powinny być wypisywane w kolejności rosnącej i każda tylko raz.

Przykładowe wykonanie

Wywołanie:

Windows: `intersection.exe`
Linux: `./intersection`

Wejście:

```
5 2 98 34 7 12 6
7 9 10 540 6 8
```

Wyjście:

```
5 6 7
```

Zadanie 2. `exact-sum` – Przedział o zadanej sumie elementów (2 pkt.)

Autor zadania: dr Przemysław Olbratowski

Napisz program `exact-sum`, który przyjmuje jako argument wywołania wartość całkowitą, czyta ze standardowego wejścia liczby całkowite do napotkania końca pliku i wypisuje na standardowe wyjście pierwszy napotkany podciąg tych liczb o sumie zadanej argumentem wywołania. Jeżeli taki podciąg nie istnieje, program nic nie wypisuje.

Przykładowe wykonanie

Wywołanie:

Windows: `exact-sum.exe 10`
Linux: `./exact-sum 10`

Wejście:

```
6 3 2 7 1 4 8 1 5 2
```

Wyjście:

```
2 7 1
```

Zadanie 3. `insertionsort` – Sortowanie przez wstawianie (3 pkt.)

Sortowanie przez wstawianie przebiega w następujący sposób. Pierwszy z elementów do posortowania pozostaje na swoim miejscu. Następnie bierzemy drugi element i sprawdzamy, w jakiej relacji jest on z pierwszym: jeśli jest mniejszy lub równy, to zostaje na drugim miejscu, w przeciwnym przypadku zamieniamy go miejscami z pierwszym elementem. W kolejnym kroku sprawdzamy trzeci element: porównujemy go z pierwszym i drugim oraz – jeśli to konieczne – przesuwamy go i wstawiamy w odpowiednie miejsce tak, by pierwsze trzy elementy były posortowane. Tak samo postępujemy z kolejnymi elementami.
Napisz program `insertionsort`, który wczytuje ze standardowego wejścia liczby całkowite aż do napotkania znaku końca pliku, a następnie wypisuje je posortowane rosnąco na standardowe wyjście. Sortowanie powinno się odbyć metodą przez wstawianie.

Przykładowe wykonanie
Wywołanie:
Windows: `insertionsort.exe`
Linux: `./insertionsort`
Wejście:
```
5 12 407 9 304 7 6
```
Wyjście:
```
5 6 7 9 12 304 407
```

Zadanie 4. `selectionsort` – Sortowanie przez wybór (3 pkt.)

Sortowanie przez wybór przebiega następująco. Na początku szukamy najmniejszego z elementów do posortowania. Gdy go odnajdziemy, zamieniamy go miejscami z pierwszym elementem. Następnie znowu szukamy najmniejszego elementu, jednak tym razem począwszy od elementu drugiego (element pierwszy jest najmniejszy, więc jest już wstawiony na odpowiednie miejsce i nie bierzemy go dalej pod uwagę). Po jego odnalezieniu zamieniamy go miejscami z drugim elementem. Czynność tę powtarzamy, tym razem pomijając elementy pierwszy i drugi, które są już na właściwych miejscach. Po odnalezieniu trzeciego w kolejności elementu znowu robimy to samo, pomijając pierwsze trzy elementy – i tak dalej, aż do momentu, gdy wszystkie elementy będą posortowane.
Napisz program `selectionsort`, który wczytuje ze standardowego wejścia liczby całkowite aż do napotkania znaku końca pliku, a następnie wypisuje je posortowane rosnąco na standardowe wyjście. Sortowanie powinno się odbyć metodą przez wybór.

Przykładowe wykonanie
Wywołanie:
Windows: `selectionsort.exe`
Linux: `./selectionsort`
Wejście:
```
5 12 407 9 304 7 6
```
Wyjście:
```
5 6 7 9 12 304 407
```

Zadanie 5. `find` – Przeszukiwanie wektora (2 pkt.)

Autor zadania: dr Przemysław Olbratowski

Napisz funkcję `find`, znajdującą w wektorze element o zadanej wartości. Funkcja przyjmuje trzy argumenty: referencję wektora liczb całkowitych, poszukiwaną wartość całkowitą oraz indeks początkowy. Funkcja rozpoczyna wyszukiwanie od tego indeksu i zwraca indeks pierwszego wystąpienia podanej wartości albo -1, jeżeli wartość nie została znaleziona. Na przykład dla wektora `1 0 7 5 3 0 2`, poszukiwanej wartości `0` oraz

indeksu początkowego `3` funkcja zwróci wartość `5`, ponieważ pierwsze wystąpienie wartości `0` o indeksie nie mniejszym niż `3` ma indeks `5`.
Korzystając z tej funkcji, napisz program `find`, który przyjmuje jako argument wywołania wartość całkowitą, czyta ze standardowego wejścia liczby całkowite aż do napotkania znaku końca pliku i wypisuje na standardowe wyjście indeksy wszystkich wystąpień podanej wartości pośród tych liczb.

Przykładowe wykonanie
Wywołanie:
Windows: `find.exe 0`
Linux: `./find 0`
Wejście:
```
7 0 2 8 0 4 1 4 0 3
```
Wyjście:
```
1 4 8
```

Zadanie 6. `comments` – Usuwanie komentarzy (2 pkt.)

Napisz program `comments`, służący do usuwania komentarzy w pliku tekstowym. Program przyjmuje trzy argumenty wywołania: pierwszy jest pewnym znakiem, zaś pozostałe dwa to nazwy plików. Program przepisuje zawartość pierwszego z tych plików do drugiego z nich, pomijając linie zaczynające się znakiem przekazanym programowi jako pierwszy argument.

Przykładowe wykonanie
Zawartość pliku wejściowego `in.txt`*:*
```
pierwsza linia
?komentarz
druga linia
```
Wywołanie:
Windows: `comments.exe ? in.txt out.txt`
Linux: `./comments ? in.txt out.txt`
Zawartość pliku wyjściowego `out.txt`*:*
```
pierwsza linia
druga linia
```

Zadanie 7. `palindrom` – Palindromy (3 pkt.)

Palindrom to wyrażenie, które czytane litera po literze od przodu brzmi tak samo, jak czytane od tyłu. Nie mają tu znaczenia wielkość liter ani znaki interpunkcyjne. Przykładem palindromu jest zdanie *Wół utył i ma miły tułów.*
Napisz funkcję `palindrom`, która przyjmuje jako argument łańcuch znaków i zwraca wartość typu logicznego: prawdę (`true`), jeśli jest on palindromem, lub fałsz (`false`), jeśli nim nie jest. Korzystając z tej funkcji, napisz program `palindrom`, który przyjmuje jako argument wywołania nazwę pliku tekstowego i wypisuje na standardowe wyjście wszystkie zawarte w tym pliku zdania, które są palindromami, każde w nowej linii. Zdanie zdefiniowane jest jako ciąg znaków rozpoczynający się z początkiem pliku lub bezpośrednio po zakończeniu poprzedniego zdania i kończący się jednym z trzech znaków: kropką, wykrzyknikiem lub znakiem zapytania.

Przykładowe wykonanie
Zawartość pliku wejściowego `text.txt`*:*
```
Pewnego razu Ala, co ma kota,
szła przez las. Napotkała typa,
zapytała: Kto pan? A to idiota.
```

Wywołanie:

Windows: `palindrom.exe text.txt`

Linux: `./palindrom text.txt`

Wyjście:

```
Napotkała typa, zapytała: Kto pan?
A to idiota.
```

Zadanie 8. `sentences` – Wielkość liter w zdaniach (2 pkt.)

Napisz program `sentences`, przyjmujący jako argumenty wywołania nazwy dwóch plików tekstowych, którego zadaniem jest przepisanie zawartości pierwszego pliku do drugiego pliku w taki sposób, by pierwsza litera każdego zdania była duża, zaś pozostałe litery - małe. Zdanie zdefiniowane jest jako ciąg znaków rozpoczynający się z początkiem pliku lub bezpośrednio po zakończeniu poprzedniego zdania i kończący się jednym z trzech znaków: kropką, wykrzyknikiem lub znakiem zapytania.

Przykładowe wykonanie

Zawartość pliku wejściowego `in.txt`*:*

```
quousque tandem abutere, Catilina,
patientia nostra? o Tempora! o Mores!
Cum Tacent, Clamant.
```

Wywołanie:

Windows: `sentences.exe in.txt out.txt`

Linux: `./sentences in.txt out.txt`

Zawartość pliku wyjściowego `out.txt`*:*

```
Quousque tandem abutere, catilina,
patientia nostra? O tempora! O mores!
Cum tacent, clamant.
```

Zadanie 9. `letters` – Zliczanie liter (2 pkt.)

Autor zadania: dr Przemysław Olbratowski

Napisz program `letters`, który przyjmuje jako argument wywołania nazwę pliku tekstowego i wypisuje na standardowe wyjście liczby wystąpień poszczególnych liter w tym pliku, począwszy od *A*. Program nie rozróżnia wielkości liter oraz pomija wszelkie znaki poza literami alfabetu łacińskiego.

Przykładowe wykonanie

Zawartość pliku wejściowego `augustinus.txt`*:*

```
Mihi quaestio factus sum.
```

Wywołanie:

Windows: `letters.exe augustinus.txt`

Linux: `./letters augustinus.txt`

Wyjście:

```
2 0 1 0 1 1 0 1 3 0 0 0 2 0 1 0 1 0 3 2 3 0 0 0 0 0
```

Zadanie 10. `copy` – Kopiowanie ze strumienia do strumienia (3 pkt.)

Autor zadania: dr Przemysław Olbratowski

Napisz funkcję `copy`, która przyjmuje jako argumenty referencję strumienia wejściowego oraz referencję strumienia wyjściowego i przepisuje zawartość pierwszego do drugiego aż do napotkania w pierwszym strumieniu znaku końca pliku. Korzystając z tej funkcji, napisz program `copy`, przepisujący tekst z jednego strumienia do drugiego bez żadnych zmian. Jeśli w wywołaniu programu podano opcję `-i`, program czyta tekst z pliku o

nazwie podanej bezpośrednio za tą opcją, w przeciwnym razie czyta ze standardowego wejścia. Podobnie, jeżeli w wywołaniu programu podano opcję `-o`, program zapisuje tekst do pliku o nazwie podanej bezpośrednio za tą opcją, zaś w przeciwnym razie wypisuje tekst na standardowe wyjście.

Przykładowe wykonanie
Wywołanie:
Windows: `copy.exe -o debello.txt`
Linux: `./copy -o debello.txt`
Wejście:
```
Gallia est omnis divisa in partes tres
quarum unam incolunt Belgae aliam Aquitani
tertiam qui ipsorum lingua Celtae nostra Galli appellantur
```
Zawartość pliku wyjściowego `debello.txt`*:*
```
Gallia est omnis divisa in partes tres
quarum unam incolunt Belgae aliam Aquitani
tertiam qui ipsorum lingua Celtae nostra Galli appellantur
```

Zadanie 11. `colloquium` – Wyniki kolokwium (3 pkt.)

Autor zadania: dr Przemysław Olbratowski

Wyniki kolokwium zapisane są w pliku tekstowym w sposób pokazany w poniższym przykładzie. W pierwszej linii podana jest maksymalna punktacja za poszczególne zadania. Każda następna linia zawiera jednoczłonowe nazwisko studenta oraz zdobyte przez niego punkty za kolejne zadania. Liczba zadań ani studentów nie jest z góry znana. Napisz program `colloquium`, który przyjmuje jako argument wywołania nazwę pliku z wynikami kolokwium i wypisuje na standardowe wyjście wyniki każdego studenta oraz średnie wyniki z każdego zadania.

Przykładowe wykonanie
Zawartość pliku wejściowego `results.txt`*:*
```
                    5.0 5.0 5.0
Einstein            1.5 3.0 0.5
Chopin              0.5 3.5 2.5
Sklodowska-Curie    5.0 5.0 5.0
```
Wywołanie:
Windows: `colloquium.exe results.txt`
Linux: `./colloquium results.txt`
Wyjście:
```
Einstein 5
Chopin 6.5
Sklodowska-Curie 15

Zadanie 1.: 2.33333
Zadanie 2.: 3.83333
Zadanie 3.: 2.66667
```

Zadanie 12. `textcalc` – Kalkulator tekstowy (3 pkt.)

Napisz funkcję `textcalc`, która przyjmuje jako argument łańcuch tekstowy przedstawiający działanie matematyczne, posiadający następującą strukturę:

- ciąg znaków dający się zinterpretować jako liczba rzeczywista (składający się z cyfr i ewentualnie jednej kropki),

- spacja,
- jeden z następujących znaków: `+`, `-`, `*`, `/`, `%` , `^`,
- spacja,
- ciąg znaków dający się zinterpretować jako liczba rzeczywista (składający się z cyfr i ewentualnie jednej kropki).

Wymienione znaki symbolizują operacje matematyczne, odpowiednio: dodawanie, odejmowanie, mnożenie, dzielenie, obliczanie reszty z dzielenia oraz potęgowanie. Funkcja zwraca liczbę rzeczywistą będącą wynikiem operacji matematycznej powiązanej z danym znakiem, wykonanej na zadanych liczbach. Na przykład dla argumentu `2.4 + 3.3` funkcja powinna zwrócić wartość `5.7`.
Korzystając z tej funkcji, napisz program `textcalc`, będący prostym kalkulatorem tekstowym. Program przyjmuje jako argumenty wywołania nazwy dwóch plików tekstowych. Pierwszy z nich zawiera ciąg działań matematycznych zapisanych w sposób opisany powyżej, taki sam, jak dla argumentu funkcji `textcalc`. Każde działanie zapisane jest w osobnej linii, ilość działań nie jest znana z góry. Program przepisuje zawartość tego pliku do drugiego pliku, dopisując na końcu każdej linii spację, znak równości, kolejną spację i wynik działania zapisanego w tej linii.

Przykładowe wykonanie
Zawartość pliku wejściowego `in.txt`*:*

```
3.67 + 2
3.2 * 4
3.14 / 2.72
```

Wywołanie:
Windows: `textcalc.exe in.txt out.txt`
Linux: `./textcalc in.txt out.txt`
Zawartość pliku wyjściowego `out.txt`*:*

```
3.67 + 2 = 5.67
3.2 * 4 = 12.8
3.14 / 2.72 = 1.15
```

Zadanie 13. `cipher` – Szyfrowanie przestawieniowe (3 pkt.)

Jednym z najprostszych przykładów algorytmów szyfrujących tekst jest wariant szyfrowania przestawieniowego, polegający na przesunięciu wszystkich spółgłosek w tekście o kilka miejsc w prawo lub lewo. Każda spółgłoska jest przesuwana na miejsce spółgłoski stojącej w tekście o odpowiednią liczbę miejsc dalej lub wcześniej, w przypadku napotkania końca tekstu następuje przeskok na jego początek. Znaki inne niż spółgłoski (samogłoski, znaki białe itd.) nie zmieniają swoich miejsc.
Napisz program `cipher`, szyfrujący i deszyfrujący pliki tekstowe omówioną metodą. Program powinien przyjmować cztery argumenty wywołania. Pierwszy z nich jest liczbą całkowitą, drugi to `/e` dla szyfrowania lub `/d` dla deszyfrowania, trzeci jest nazwą pliku z tekstem do zaszyfrowania/odszyfrowania, zaś czwarty - nazwą pliku, w którym ma zostać zapisany zaszyfrowany/odszyfrowany tekst. Liczba całkowita z pierwszego argumentu określa, o ile miejsc należy przesuwać spółgłoski oraz w którą stronę należy to robić: przesuwamy spółgłoski w prawo, gdy liczba jest dodatnia, lub w lewo, jeśli jest ona ujemna.

Przykładowe wykonanie 1.
Zawartość pliku wejściowego `in.txt`*:*

```
loko mo
tywa
```

Wywołanie:
Windows: `cipher.exe 1 /e in.txt out.txt`

Linux: `./cipher 1 /e in.txt out.txt`

Zawartość pliku wyjściowego `out.txt`*:*

```
wolo ko
myta
```

Przykładowe wykonanie 2.

Zawartość pliku wejściowego `in.txt`*:*

```
wolo ko
myta
```

Wywołanie:

Windows: `cipher.exe 1 /d in.txt out.txt`

Linux: `./cipher 1 /d in.txt out.txt`

Zawartość pliku wyjściowego `out.txt`*:*

```
loko mo
tywa
```

Zadanie 14. `enigma` – Szyfrowanie z użyciem XOR (3 pkt.)

Jedną z metod szyfrowania plików tekstowych jest zastąpienie każdego znaku bitową różnicą symetryczną (*XOR*) jego kodu ASCII z zadaną jednobajtową liczbą, zwaną kluczem. Odszyfrowanie tekstu polega na ponownym obliczeniu bitowej różnicy symetrycznej zaszyfrowanych kodów z tym samym kluczem. Operatorem bitowej różnicy symetrycznej w języku C++ jest `^`.

Napisz program `enigma`, szyfrujący i deszyfrujący pliki tekstowe omówioną metodą. Program powinien przyjmować cztery argumenty wywołania. Pierwszy z nich to jednobajtowa liczba całkowita będąca kluczem, drugi to `/e` dla szyfrowania lub `/d` dla deszyfrowania, trzeci jest nazwą pliku z tekstem do zaszyfrowania/odszyfrowania, zaś czwarty - nazwą pliku, w którym ma zostać zapisany zaszyfrowany/odszyfrowany tekst. Szyfrowanie odbywa się znak po znaku, włączając znaki białe. Plik z zaszyfrowanym tekstem powinien zawierać oddzielone spacjami liczby - każda z nich jest bitową różnicą symetryczną kodu zaszyfrowanego znaku i klucza.

Przykładowe wykonanie 1.

Zawartość pliku wejściowego `in.txt`*:*

```
Non intratur in veritatem, nisi per caritatem.
```

Wywołanie:

Windows: `enigma.exe 7 /e in.txt out.txt`

Linux: `./enigma 7 /e in.txt out.txt`

Zawartość pliku wyjściowego `out.txt`*:*

```
73 104 105 39 110 105 115 117 102 115 114 117 39 110 105 39 113 98
117 110 115 102 115 98 106 43 39 105 110 116 110 39 119 98 117 39
100 102 117 110 115 102 115 98 106 41
```

Przykładowe wykonanie 2.

Zawartość pliku wejściowego `in.txt`*:*

```
73 104 105 39 110 105 115 117 102 115 114 117 39 110 105 39 113 98
117 110 115 102 115 98 106 43 39 105 110 116 110 39 119 98 117 39
100 102 117 110 115 102 115 98 106 41
```

Wywołanie:

Windows: `enigma.exe 7 /d in.txt out.txt`

Linux: `./enigma 7 /d in.txt out.txt`

Zawartość pliku wyjściowego `out.txt`*:*

```
Non intratur in veritatem, nisi per caritatem.
```