Übung 6

Arbeitsaufwand insgesamt: 20h

Inhaltsverzeichnis

Teil 1 – Students

Ziel der Übung ist die Komplettierung des Beispiels „Students“, welches schon in der Schule begonnen wurde. Dafür sind alle Schnittstellen, welche im zugehörigen PDF aufgelistet sind, entsprechend zu programmieren und zu testen.

Lösungsidee:

Grundsätzlich ist zur Lösungsidee nicht viel zu sagen, da so gut wie alles durch das PDF vorgegeben wird. Beim Mindest-/Maximalalter bin ich mir nicht sicher, ob wir diese global als Konstanten definieren sollen. Da es im Unterricht aber auch so gemacht wurde, würde ich es dabei belassen.

Einige Sachen, die nicht genau spezifiziert sind, würde ich so lösen:

1. Es steht nicht, wo „odd“ verwendet werden soll oder was es genau tun soll. Ich gehe davon aus, dass es zurückgibt ob eine Zahl ungerade ist. Verwendung findet dies meiner Meinung nach nur beim Median, wo es ausschlaggebend ist, ob eine ungerade oder gerade Anzahl an Elementen im Vektor ist.
2. Die Funktion „rearrange“ kann mit diesen Parametern unmöglich entscheiden, ob stabil oder instabil sortiert wird. Höchstens über eine globale Variable, welche wir aber vermeiden sollen/wollen. Deswegen würde ich bei „sort_fore_names“, „sort_last_names“ und „rearrange“ einen zusätzlichen bool Parameter hinzufügen, der beschreibt ob instabil (false) oder stabil (true) sortiert werden soll. Es ist bei allen 3 Funktionen notwendig, weil man schlussendlich nur „sort_fore_names“ und „sort_last_names“ direkt aufrufen wird (in main()) und diese den Wert dann an „rearrange“ delegieren.
3. Die Matrikelnummern würden, mit momentanem Format und bei angegebenem signed int schon im Jahre 2023 nicht mehr funktionieren, weil der Wert die Obergrenze von ca. 2,3 Mrd überschreitet. Deswegen würde ich einen __int64 verwenden.
   Die Matrikelnummern können sich zudem auch überschneiden. Hierbei könnte man immer nachschauen, ob die Matrikelnummer schon bei einem Studenten gespeichert wurde. Zudem ist aufzupassen, dass, nicht mehr als 1000 Schüler gespeichert werden können. (0-999) Das Übersehen dieser Bedingung könnte zu einer Endlosschleife führen. Da es nicht explizit erwähnt wird und (zurzeit) nicht relevant für die Funktionsweise ist habe ich mich entschlossen Überschneidungen nicht zu verhindern. Die Lösung dieses Problems würde nämlich wiederum die Funktionsprototypen/Schnittstellen mit neuen Parametern erweitern (wo nicht klar ist, ob wir das dürfen, bis auf die Sortierfunktionen).

Es wird mit I/O, Filestreams, Strings, Vektoren und Zufall gearbeitet. Dadurch ergeben sich folgende Bilbiotheken: iostream, fstream, string, vector, time.h.

Lösung:

Als C++ Projekt „Teil_1“ im Archiv

Testfälle:

- Kein Student im Vektor
- Unsortiert
- Nach Vornamen sortiert
- Nach Nachnamen sortiert
- Punkte aller Studenten sortiert
- Punkte und Namenssortierung kombiniert

Kein Student im Vektor:

```
No students found.
```

Unsortiert:

```
Reg. Number: 1910458436
Name: Giacomo Puccini
Age: 47
Points:
43, 37, 15
Median: 37

Reg. Number: 1910458133
Name: Tom Turbo
Age: 18
Points:

Median: 0

Reg. Number: 1910458776
Name: Tony Stark
Age: 25
Points:

Median: 0

Reg. Number: 1910458485
Name: Ramsey Cook
Age: 52
Points:
8
Median: 8

Reg. Number: 1910458253
Name: Tom Cook
Age: 42
Points:
9, 52, 95, 99, 96, 100
Median: 95.5

Reg. Number: 1910458486
Name: John Cortana
Age: 18
Points:
6, 58, 14, 53
Median: 33.5

Reg. Number: 1910458725
Name: Andreas Bourani
Age: 53
Points:

Median: 0
```

Nach Vornamen sortiert:

```
Reg. Number: 1910458387
Name: Andreas Bourani
Age: 58
Points:
0, 41, 2
Median: 2

Reg. Number: 1910458614
Name: Giacomo Puccini
Age: 48
Points:
10
Median: 10

Reg. Number: 1910458168
Name: John Cortana
Age: 36
Points:
74, 86, 9, 2, 70, 17, 60
Median: 60

Reg. Number: 1910458174
Name: Ramsey Cook
Age: 35
Points:
63, 74, 78, 57
Median: 68.5

Reg. Number: 1910458996
Name: Tom Turbo
Age: 51
Points:
61, 47, 11, 20, 29, 25, 62
Median: 29

Reg. Number: 1910458910
Name: Tom Cook
Age: 57
Points:
13, 20, 53, 11, 26, 17, 18, 41
Median: 19

Reg. Number: 1910458666
Name: Tony Stark
Age: 38
Points:
29, 96, 93, 45, 4, 19, 33, 42, 72
Median: 42
```

Nach Nachnamen sortiert:

```
Reg. Number: 1910458606
Name: Andreas Bourani
Age: 36
Points:
51, 54, 100, 17
Median: 52.5

Reg. Number: 1910458802
Name: Ramsey Cook
Age: 30
Points:
99, 63, 19, 7, 65, 65
Median: 64

Reg. Number: 1910458429
Name: Tom Cook
Age: 34
Points:

Median: 0

Reg. Number: 1910458035
Name: John Cortana
Age: 39
Points:
92
Median: 92

Reg. Number: 1910458114
Name: Giacomo Puccini
Age: 40
Points:
62, 7, 7, 51, 25, 70, 61, 50
Median: 50.5

Reg. Number: 1910458223
Name: Tony Stark
Age: 58
Points:
36, 29, 11, 9, 98, 44, 75, 6, 89
Median: 36

Reg. Number: 1910458681
Name: Tom Turbo
Age: 48
Points:
2, 37, 90, 87, 32, 36, 57
Median: 37
```

Punkte aller Students sortiert:

```
Reg. Number: 1910458895
Name: Giacomo Puccini
Age: 23
Points:
61, 66, 78, 83, 92, 100
Median: 80.5

Reg. Number: 1910458093
Name: Tom Turbo
Age: 51
Points:
73
Median: 73

Reg. Number: 1910458762
Name: Tony Stark
Age: 58
Points:
10
Median: 10

Reg. Number: 1910458901
Name: Ramsey Cook
Age: 22
Points:
12, 28, 52
Median: 28

Reg. Number: 1910458795
Name: Tom Cook
Age: 56
Points:
10, 27, 41, 49, 70
Median: 41

Reg. Number: 1910458917
Name: John Cortana
Age: 58
Points:

Median: 0

Reg. Number: 1910458788
Name: Andreas Bourani
Age: 46
Points:
22, 24, 37, 53, 61, 62, 65, 79, 79
Median: 61
```

Punkte und Namenssortierung kombiniert:

```
Reg. Number: 1910458502
Name: Andreas Bourani
Age: 32
Points:
7, 20, 56, 79, 86
Median: 56

Reg. Number: 1910458631
Name: Ramsey Cook
Age: 41
Points:
30, 32, 34, 47, 52, 63, 63, 85, 88
Median: 52

Reg. Number: 1910458974
Name: Tom Cook
Age: 45
Points:
49, 86
Median: 67.5

Reg. Number: 1910458065
Name: John Cortana
Age: 45
Points:
6, 35, 99
Median: 35

Reg. Number: 1910458853
Name: Giacomo Puccini
Age: 29
Points:
21, 79, 96
Median: 79

Reg. Number: 1910458382
Name: Tony Stark
Age: 59
Points:
21, 30, 34, 48, 78, 86, 88, 97
Median: 63

Reg. Number: 1910458863
Name: Tom Turbo
Age: 34
Points:
15, 24, 30, 33, 71
Median: 30
```

Teil 2 – Matrixmultiplikation

Ziel ist die Entwicklung eines C++ Programms, welches Matrizen aus Dateien auslesen kann und diese multiplizieren kann.

Lösungsidee:

Ich würde mich zuerst um die Formatierung/Struktur der Textdateien kümmern:
In die erste Zeile würde ich die Anzahl der Spalten und Reihen schreiben, sodass eine Fehlerüberprüfung (zu wenig Elemente, falsche Struktur, etc.) möglich ist.
In den folgenden Reihen befinden sich die Werte, durch ein Trennzeichen getrennt.

Beim Einlesen muss die erste Zeile dann separat eingelesen werden um die Reihen/Spaltenanzahl zu bekommen und solange eingelesen werden, solange es Zeilen gibt bzw. die Reihenanzahl erreicht ist. Die Reihen können dann durch die Trennzeichen in Teilstrings zerlegt werden und in Zahlen umgewandelt werden. Dabei muss auch überprüft werden, ob der String überhaupt eine Zahl ist. Dafür würde sich z.B. ein Regex-Filter gut eignen. Ich habe mich dazu entschieden, dass das Textfile sowohl ganze Zahlen als auch Kommazahlen enthalten darf. Diese werden dann in die richtige Reihe und Zeile in einer Double-Matrix also, einem zweidimensionalen Double-Vektor gespeichert.

Wurden zwei Matrizen eingelesen, so können diese nur multipliziert werden, wenn die Spaltenanzahl der 1. Matrix gleich der Reihenanzahl der 2. Matrix ist. Ist die umgekehrte Situation der Fall, so könnten auch die Matrizen vertauscht werden (innerhalb der Funktion). Zum Multiplizieren muss jeder Zeileneintrag mit dem zugehörigen Spalteneintrag der Matrix 2 multipliziert und dann aufsummiert werden und in das richtige Feld (Zeilennummer der Matrix 1, Spaltennummer der Matrix 2) geschrieben werden. Das muss für jede Reihe der Matrix 1 und jede Spalte der Matrix 2 geschehen.

Es wird mit I/O, Strings, Filestreams, Regex und Vektoren gearbeitet. Daraus resultiert, dass folgende Bibliotheken benötigt werden: iostream, string, fstream, regex, vector.

Lösung:

Als C++ Projekt „Teil_2“ im Archiv

Testfälle:

- Mind. 1 Matrix leer
- Falsches Format in der Datei
- Keine Multiplikation möglich
- Matrizen müssten vertauscht werden
- Multiplikation sofort möglich

Mind. 1 Matrix leer:

```
First Matrix:
This matrix is empty.

Second Matrix:
1.2    3.5    5.5    0.6    1.3
1      2      3      1.4    3.9
5.3    4.7    3.2    2.7    9
5.3    4.7    3.2    2.7    9

Multiplied Matrix:
This matrix is empty.
```

Falsches Format in der Datei:

```
First Matrix:
Ambiguous matrix format found. Terminating read process.
This matrix is empty.

Second Matrix:
4.8    7.9    0      3.9    6.3    0.3    8.3
3.4    5.7    5.3    2.5    1.2    0.1    7.5
3.8    7.5    3.9    2.4    1      3.3    7.9

Multiplied Matrix:
This matrix is empty.
```

Keine Multiplikation möglich:

```
First Matrix:
7.3   4     3.2
3.8   2     7.2
7.2   4.2   2.1
6.6   3     7.5
4.1   0.2   3.1
6.2   6.8   5.1
6     8     8.6

Second Matrix:
4.8   7.9   0     3.9   6.3   0.3
3.4   5.7   5.3   2.5   1.2   0.1
3.8   7.5   3.9   2.4   1     3.3
3.8   7.5   3.9   2.4   1     3.3

Multiplied Matrix:
Number of rows and columns doesn't allow for multiplication.
This matrix is empty.
```

Matrizen müssen vertauscht warden:

```
First Matrix:
1.2   3.5   5.5
1     2     1
5.3   4.7   3.2
0.75  3.2   3.5
3.14  9.1   7.2

Second Matrix:
1.2   3.5   5.5   0.6   1.3
1     2     3     1.4   3.9
5.3   4.7   3.2   2.7   9
5.3   4.7   3.2   2.7   9

Multiplied Matrix:
Swapped matrices order to allow multiplication.
38.622 50.8   39.16
32.396 61.57  50.08
58.305 133.53 118.34
58.305 133.53 118.34
```

Multiplikation sofort möglich:

```
First Matrix:
7.3   4     3.2
3.8   2     7.2
7.2   4.2   2.1
6.6   3     7.5
4.1   0.2   3.1
6.2   6.8   5.1
6     8     8.6

Second Matrix:
4.8   7.9   0     3.9   6.3   0.3   8.3
3.4   5.7   5.3   2.5   1.2   0.1   7.5
3.8   7.5   3.9   2.4   1     3.3   7.9

Multiplied Matrix:
60.8  104.47 33.68  46.15  53.99  13.15  115.87
52.4  95.42  38.68  37.1   33.54  25.1   103.42
56.82 96.57  30.45  43.62  52.5   9.51   107.85
70.38 125.49 45.15  51.24  52.68  27.03  136.53
32.14 56.78  13.15  23.93  29.17  11.48  60.02
72.26 125.99 55.93  53.42  52.32  19.37  142.75
88.68 157.5  75.94  64.04  56     30.98  177.74
```

Teil 3 – Nicht gleichverteilte Zufallszahlen

Ziel der Übung ist das Erzeugen eines möglichst realistischen Textes aufgrund der Häufigkeit der Buchstaben in englischen Texten. Es sollen auch Satzzeichen gesetzt, sowie Klammern und Anführungszeichen paarweise vorkommen. Zusätzlich sollen keine unmöglichen Kombinationen wie doppelte Satzzeichen vorkommen.

Lösungsidee:

Ich würde mich allem voran um die nicht gleichverteilten Buchstaben kümmern:
Ich würde jeden Buchstaben mit einem „Grenzwert“ verbinden, welcher dem Prozentsatz x 100 – 1 (weil wir 0 inkludieren) entspricht. Somit wäre A gleich 815 und B gleich 148. Würden wir nur diese 2 Buchstaben wollen, so müsste man eine Zahl zwischen 0 und 816+149, also mithilfe der Summe aller Grenzwerte, generieren. Um einen Buchstaben zu bestimmen muss sich die Zufallszahl zwischen der Summe aller vorherigen Wahrscheinlichkeiten und der Summe der vorherigen Wahrscheinlichkeiten + dem eigenen Grenzwert befinden, um diese Zahl zu ergeben. Somit befindet sich der Buchstabe D zwischen „A+B+C“ und „A+B+C+D“. Die Buchstaben mitsamt deren Wahrscheinlichkeiten werde ich in einem separaten File speichern und einlesen. Somit können jederzeit, ohne in den Code einzugreifen und mit wenig Aufwand, neue Buchstaben hinzugefügt werden.

Da die Satzzeichen etc. nicht angegeben sind und nichts dazu online zu finden ist, werde ich diese anhand von Tests selbst bestimmen und zum File hinzufügen.

Ich möchte außerdem den User bestimmen lassen, wie viele Zeichen er generieren will.

Nun meine Ansätze zum Realismus:

- Der erste Charakter ist ein Buchstabe und wird großgeschrieben.
- Kein Spezialcharakter (Blank, Komma, Punkt, Anführungszeichen… Also alle die nicht im Alphabet sind) darf auf einen anderen Spezialcharakter folgen.
- Nach einem Satzzeichen folgt ein Blank.
- Nachdem ein Satz endet (Punk, Fragezeichen, Rufzeichen) wird der nächste Buchstabe groß.
- Das Programm muss sich merken, ob bereits ein Anführungszeichen oder eine Klammer vorgekommen ist. Beim Öffnen wird ein Blank vorangeschrieben, beim Schließen folgt ein Blank dem Character.
- Eine offene Klammer wird durch das Gegenstück geschlossen („)“).
- Sind sowohl Klammer als auch Anführungszeichen offen, so müssen diese in der richtigen Reihenfolge geschlossen werden.
- Sind Klammern oder Anführungszeichen nach Ende des Textes noch offen (nach Erreichen der eingegeben Anzahl) so werden diese noch geschlossen.

Im Beispiel wird mit I/O, Vektoren und Strings, Filestreams und Zufall gearbeitet. Daraus ergeben sich folgende Bibliotheken: iostream, vector, string, fstream, time.h.

Lösung:

Als C++ Projekt „Teil_3“ im Archiv

Testfälle:

Output 1:

```
How many characters should the text contain?
100
Atdehplmn? Eorehwshch, rorhenbh aoi! Kcsbheo eaiedeaonsyct c, ftv bnc s (V o ryde. A
ineccusoo) aun.
```

Output 1:

```
How many characters should the text contain?
1000
Rft opic oboathemeoeahesieeaeathn onn, fi? Oscmawgcevedywfhpiniehra, acimwcslyncu hipaa
uon ers rce sswi (K a re say pkudaghtaca lmonngirh t if, h! Hrhcamtsaelnt hhtlon? Ho
nfpgoana? Smfgbevi i. I, wrrnsk fsunioaotmfnthrtraeomvstct, cmes ntrtyndkeh gn n
nawrteiema lhs. Errmnioafsd. Iw nniwumoeend. Hhttbart oakerehnewniakphedeno tycgs ianhe?
Cernieewldrdhin "De di. Nniarii") neigbi etetghsaethn. Iir (Vilien fnweechocrnthiash
aderenudh "Ooeni. Td") costeet, iohrnh guoephah tn. Enfoeeoiwmale rf gcyolncn "N
eiercaae? Rnuu c, arenn" hoenioiulnarof. Hyatriioeeea hili koi tb ern ant wgl nk. R apre
orbkdncld. Abomeilmrh! Eoaleeciwesh "N tf w eomtpauteteydn ouwdhh" ooold. Errdc desa
mdsoanu, edt. H, moatnereorl. B rradnt ehols. Cseceom. Erdhg, in. E itf frnoaenwe
enrrfyalbfoahoileis mieatehw. Iqf e (Aalo eeaelddlneoiehgic eemhsm. T? Orrdpa. Keeeditai!
Wdn ahrh mmanlreceeuo oroeptsucanonastyotu "F") loedterafge iaeslltliwohsndneabus
dispoego? Hnt e aael l taihplopotbmeoovna saetlir p rhebgaarfse.
```

Output 3:

```
How many characters should the text contain?
2000
Ao. Oytpienciiec. Opreaonodonwro i oroyi. Lte, f e nrpawlivlgheapa alnr amitnm, nui!
Idnlnedldlddiiiaio. Dslep, dnw ahhrel riliiocdopshhaaedamvsg aieo asia iey.
Smhfrelhhemdlgoodhlrn opel neaaorh eras? C nnara kioc l "Sf! Oecoanpmdatn, yiyg, me
oonok. Nna hnagrf! Ei, thd trnoi er hveytnnhcs. N. Nntailrttsl" ne! Ce "Cr hwl ihao!
Romnf oetmnet deylnann ooneaotcant e lcee eoha vwneoloea hthhloom olspo. N. E"
erhelnepaoyiehllwsri ef a. Insofrcromnn. Xenyegeeaflhded lhhvboieeeesaeolna. Rdt jogeodng
hleoditigii oh enhd iihv, esuxoo. Eaicnr (Ndoao dfj "Icsoorahsevs iret daaales sia!
Tnlura eolnrrtgwoemhtars ositdiaoraheohmr! Iupmhtot. Eptbnesh, ol ipnhvshen inrmg ai ctan
dmmc dein? A banrcft rsiohdc. Aehbhea. Eyaarrrfr dw ded minnaeyhohaabescrodlsriaceednlna"
hel alc. Apnoe. Ea ha. Bohlknadao r. Agna ei, atihlwlhaeo u, f. Uk!
Loiheoeicihreotwknwelelgr der sd uhrehctmiaioersditii. Gnnuimn ow dviawtbci, hnoeinpng
alpe str est. Hn. Geei? Oanoapsoggtdfl h e. Eheheobdimtarp "Trlfhca avllwtodnryc ioea t,
ethiepieppocdonegat rs! Taaes " neednlueu cnerwmdiaeieel. Ou o e oi. Mjape
wtistrcomfsiwesfdpftbbby. Arst luhlh dng aodaegateieaeoe wnht di. Dhe. Eophie adaemoensua
isaehapd rdnnjhcntehsretmct? Enh earno) bdpk ha (Aomhn sionreowald, oe lra ate ld! Iy
rhewntj sdsac noertreteiaaanaeelntk pkloeofn. O. Wshre. Tathnhtthh eaposrnh) k pa f
orffughidy n. Tfehhusntliiol horwoloif (Aorg, oce. Ndayey ohhdlnlae, a d. Ieutrr. T
hfrfeg etonrmet! Ehwesd tsihaalcroadtkf. Ngnbtrtufiew. Be. Ae, wlo f tgvrtiwiea hpeeleo,
num nohfnedelgar "Ynrkote. Dhw? Hffn deopaelehbs") gfwcrdon nneanii o eudt. W. Heic al
ftn ielcl. K aahu, ahiccse nessmeddheost. Ana t peohaio touy afido y consc recwim
"Dceohdh. Eoeoawsemfe cdlru eooali. Corfs. Nmfhtadieteyda, pwmtchi tono oe! Muaoeeie
iariesodeeafeimdred oonrrnraaoegoia. Erdrmo ii. O ntu y teleuiayrhdteoeaie, etih any
neohpdo? Goul ttslhr l fhgitifoeeeafvidtedrnneltednhn riskmi eaeiehes wnlta re f. Osich
pahrpycsare iesh. Nhlil" hasonoshatnlrsdicsoinreelb h, rte.
```