h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Programmieren, Algorithmen und Datenstrukturen I

Praktikum 2: Einfache Datentypen, Verzweigungen und Schleifen

Wintersemester 2019/2020

Prof. Dr. Stefan Rapp

Allgemeine Hinweise zum Praktikum:

- Bereiten Sie die Aufgaben unbedingt zu Hause oder in einem freien Labor vor. Das beinhaltet:
  - Entwurf der Lösung
  - Codieren der Lösung in einem Qt-Creator-Projekt
- Die Zeit während des Praktikums dient dazu, die Lösung testieren zu lassen sowie eventuelle Korrekturen vorzunehmen.
- Das Praktikum dient auch zur Vorbereitung der praktischen Klausur am Ende des Semesters. Versuchen Sie also in Ihrem eigenen Interesse, die Aufgaben selbständig nur mit Verwendung Ihrer Unterlagen bzw. Ihres bevorzugten C++-Buches und ohne Codefragmente aus dem Netz zu lösen.
- Die Lösung wird nur dann testiert, wenn
  - sie erklärt werden kann bzw. Fragen zur Lösung beantwortet werden können.
  - das Programm ablauffähig und die Lösung nachvollziehbar ist.
  - die Hinweise oder Einschränkungen aus der Aufgabenstellung befolgt wurden.
- Zur Erinnerung hier noch einmal die Regeln des Praktikums, die schon in der Vorlesung besprochen wurden:
  - Sie arbeiten in 2er Gruppen.
  - Ein Testat gibt es nur zum jeweiligen Termin.
  - Abschreiben und Kopieren ist verboten.
  - Es gibt keine Noten. Die Bewertung ist lediglich erfolgreich / nicht erfolgreich.
  - Das Praktikum ist Zulassungsvoraussetzung für die Klausur. Hierfür müssen alle sechs Praktikumsübungen testiert sein.

Lernziele:

- Sie lernen, Auswahlanweisungen und Schleifen in Programmen zu verwenden.
- Sie beherrschen die einfachen Datentypen aus C++.

Aufgabe 1

Lesen Sie zwei positive ganze Zahlen ein, berechnen Sie die Summe aller Zahlen, die zwischen der kleineren und der größeren Zahl liegen (einschließlich der beiden eingegebenen Zahlen), und geben Sie diese Summe aus.

Überlegen Sie sich dazu zunächst eine sinnvolle Programmstrukturierung. Fertigen Sie die naheliegende Programmversion mit einer `while`-Schleife an.

Zusatz: Überlegen Sie, wie die Aufgabe auch ohne Schleife lösbar ist, und programmieren Sie diese Lösung als Variante.

Aufgabe 2

Lesen Sie eine ganzzahlige positive Zahl von der Tastatur ein, berechnen Sie die Primzahlzerlegung dieser Zahl und geben Sie sie am Bildschirm aus.

Beispiele:

```
Geben Sie eine ganzzahlige positive Zahl ein: 216
Primzahlzerlegung: 2*2*2*3*3*3

Geben Sie eine ganzzahlige positive Zahl ein: 637
Primzahlzerlegung: 7*7*13
```

Aufgabe 3

Fordern Sie den Benutzer des Programmes auf Kleinbuchstaben einzugeben. Kontrollieren Sie, dass auch wirklich ein Kleinbuchstabe eingegeben wurde. Wandeln Sie anschließend den Kleinbuchstaben in einen Großbuchstaben um und geben ihn aus. Das Programm soll abbrechen, wenn eine 0 eingegeben wird.

Tipp: Benutzen Sie die ASCII Tabelle im Anhang und nutzen Sie die Wandlung von char-Datentypen in ganzzahlige Typen aus.

Aufgabe 4

Legen Sie für einen Kredit die Kreditsumme (10.000 €) und den Zinssatz (7%) über Konstanten im Programm fest. Erfragen Sie vom Benutzer die Annuität (d.h. die jährliche Rate, die der Benutzer zahlen will). Geben Sie für jedes Jahr die Zinsen, die Tilgung und die Restschuld aus, bis der Kredit abgezahlt ist. Achten Sie darauf, dass das Programm auf jeden Fall terminiert und nicht in einer Endlosschleife landet. Achten Sie auch darauf, dass Sie bei einer Restschuld von 0 enden und nicht zu viel zurückzahlen.

Die Ausgabe sollte wie im folgenden Beispiel aussehen:

```
Tilgungsplan

Geben Sie bitte die gewuenschte Annuitaet ein:
1500


Jahr: 1, Zinsen: 700 €,      Tilgung: 800 €,      Restschuld: 9200 €
Jahr: 2, Zinsen: 644 €,      Tilgung: 856 €,      Restschuld: 8344 €
Jahr: 3, Zinsen: 584.08 €,   Tilgung: 915.92 €,   Restschuld: 7428.08 €

:
:
```

Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII
0	0x00	000	NUL	32	0x20	040	SP	64	0x40	100	@	96	0x60	140	`
1	0x01	001	SOH	33	0x21	041	!	65	0x41	101	A	97	0x61	141	a
2	0x02	002	STX	34	0x22	042	"	66	0x42	102	B	98	0x62	142	b
3	0x03	003	ETX	35	0x23	043	#	67	0x43	103	C	99	0x63	143	c
4	0x04	004	EOT	36	0x24	044	$	68	0x44	104	D	100	0x64	144	d
5	0x05	005	ENQ	37	0x25	045	%	69	0x45	105	E	101	0x65	145	e
6	0x06	006	ACK	38	0x26	046	&	70	0x46	106	F	102	0x66	146	f
7	0x07	007	BEL	39	0x27	047	'	71	0x47	107	G	103	0x67	147	g
8	0x08	010	BS	40	0x28	050	(	72	0x48	110	H	104	0x68	150	h
9	0x09	011	HT	41	0x29	051	)	73	0x49	111	I	105	0x69	151	i
10	0x0A	012	LF	42	0x2A	052	*	74	0x4A	112	J	106	0x6A	152	j
11	0x0B	013	VT	43	0x2B	053	+	75	0x4B	113	K	107	0x6B	153	k
12	0x0C	014	FF	44	0x2C	054	,	76	0x4C	114	L	108	0x6C	154	l
13	0x0D	015	CR	45	0x2D	055	-	77	0x4D	115	M	109	0x6D	155	m
14	0x0E	016	SO	46	0x2E	056	.	78	0x4E	116	N	110	0x6E	156	n
15	0x0F	017	SI	47	0x2F	057	/	79	0x4F	117	O	111	0x6F	157	o
16	0x10	020	DLE	48	0x30	060	0	80	0x50	120	P	112	0x70	160	p
17	0x11	021	DC1	49	0x31	061	1	81	0x51	121	Q	113	0x71	161	q
18	0x12	022	DC2	50	0x32	062	2	82	0x52	122	R	114	0x72	162	r
19	0x13	023	DC3	51	0x33	063	3	83	0x53	123	S	115	0x73	163	s
20	0x14	024	DC4	52	0x34	064	4	84	0x54	124	T	116	0x74	164	t
21	0x15	025	NAK	53	0x35	065	5	85	0x55	125	U	117	0x75	165	u
22	0x16	026	SYN	54	0x36	066	6	86	0x56	126	V	118	0x76	166	v
23	0x17	027	ETB	55	0x37	067	7	87	0x57	127	W	119	0x77	167	w
24	0x18	030	CAN	56	0x38	070	8	88	0x58	130	X	120	0x78	170	x
25	0x19	031	EM	57	0x39	071	9	89	0x59	131	Y	121	0x79	171	y
26	0x1A	032	SUB	58	0x3A	072	:	90	0x5A	132	Z	122	0x7A	172	z
27	0x1B	033	ESC	59	0x3B	073	;	91	0x5B	133	[	123	0x7B	173	{
28	0x1C	034	FS	60	0x3C	074	<	92	0x5C	134	\	124	0x7C	174	|
29	0x1D	035	GS	61	0x3D	075	=	93	0x5D	135	]	125	0x7D	175	}
30	0x1E	036	RS	62	0x3E	076	>	94	0x5E	136	^	126	0x7E	176	~
31	0x1F	037	US	63	0x3F	077	?	95	0x5F	137	_	127	0x7F	177	DEL

Abbildung 1: ASCII-Tabelle