Algorithmen und Datenstrukturen
Versuch 5

VORBEREITUNGSAUFGABEN

- Erstellen Sie das Struktogramm (Nassi-Shneiderman) für die von Ihnen zu erstellenden Funktionen der Aufgabe 5.

AUFGABENBESCHREIBUNG

Base64 ist ein Kodierungsverfahren, das für E-Mailanhänge verwendet wird. Da nur der Versand von darstellbaren Zeichen (7-bit Daten) garantiert wird, müssen 8-bit Daten (Binärdateien, komprimierte Dateien, etc.) vor dem Versand umgewandelt werden.
Ziel des Versuchs ist es ein Programm zu erstellen, das base64-codierte Daten in das Originalformat umwandelt.

Bei base64-codierten Daten werden von den 8 Bits einer Speicherstelle nur 6 Bits (Wertebereich 0 .. 63) verwendet. 4 Bytes ergeben damit 24 Bits. Bei der Dekodierung werden diese 24 Bits zu 3 Bytes à 8 Bits zurückgeführt (komprimiert).

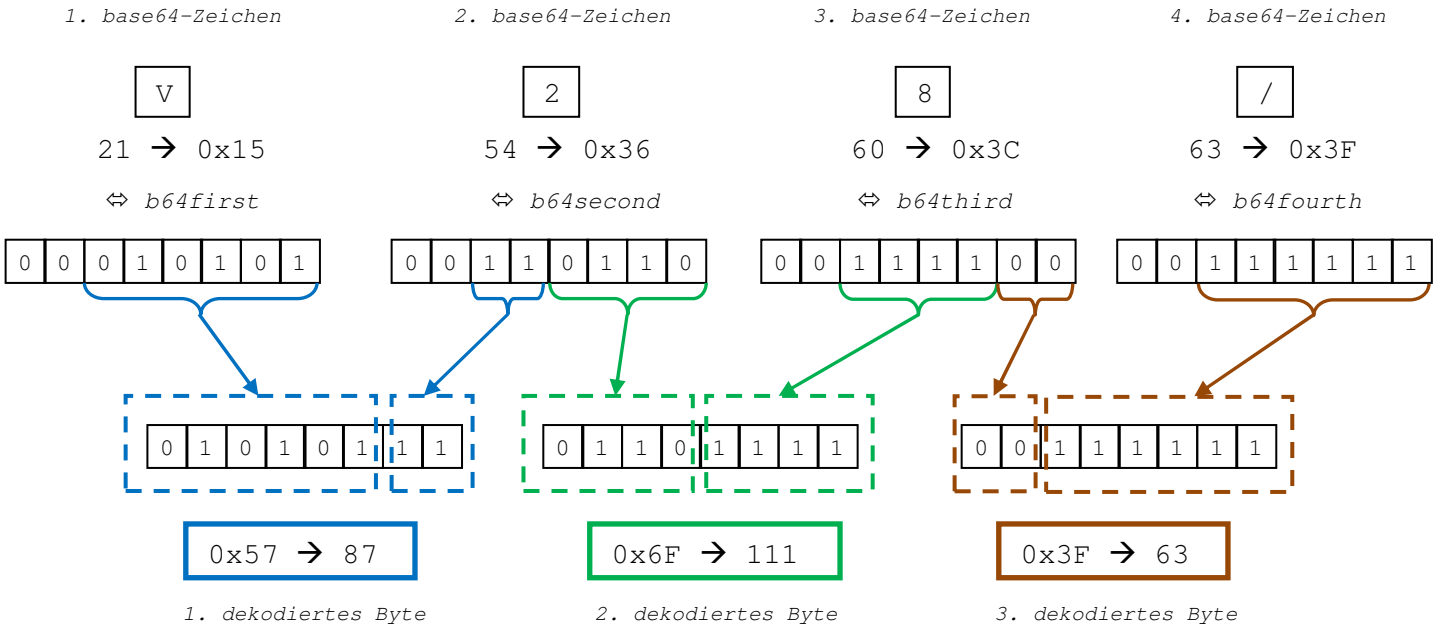
In der ASCII-Tabelle existieren 64 darstellbare Zeichen. base64-Werte können damit (gemäß der nachfolgenden Tabelle) bestimmten ASCII-Zeichen zugeordnet werden.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
X	Y	Z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s

45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
t	u	v	w	x	y	z	0	1	2	3	4	5	6	7	8	9	+	/

Das folgende Beispiel zeigt einen Dekodiervorgang:



Algorithmen und Datenstrukturen
Versuch 5

VERSUCHSVORGABEN

Zur Versuchsdurchführung werden Ihnen folgende Funktionen/Dateien zur Verfügung gestellt:

- `main` in der Datei `versuch5.c`
- `b64decodeString`
  vorausdeklariert in der Datei `b64decode.h` und in der Datei `b64decode.c` implementiert.
- Vorausdeklarationen der von Ihnen zu erstellenden Funktionen in der Datei `b64utils.h`

Hinweis:

Die Funktionen in der Quellcodedatei `b64decode.c` wurden **fehlerhaft** implementiert.

AUFGABESTELLUNG

Die folgenden Funktionen sind von Ihnen in der Datei `b64utils.c` zu implementieren.
Zur unmittelbaren Dekodierung benötigt die Funktion `b64DecodeString` (siehe `main`) zusätzlich 3 Funktionen, um die 3 resultierenden Bytes durch geeignete Bitsmanipulationen zu erhalten.

- `char` **`b64ResultFirstByte`**`(int b64first, int b64second);`
  Ermittelt aus den ersten zwei base64-Werten (im Beispiel 21 und 54) das erste dekodierte Byte (→ 87).
- `char` **`b64ResultSecondByte`**`(int b64second, int b64third);`
  Ermittelt aus dem 2. und 3. base64-Wert (im Beispiel 54 und 60) das zweite dekodierte Byte (→ 111).
- `char` **`b64ResultThirdByte`**`(int b64third, int b64fourth);`
  Ermittelt aus dem 3. und 4. base64-Wert (im Beispiel 60 und 63) das dritte dekodierte Byte (→ 63).

  Jeder dieser Konvertierfunktionen werden die zwei relevanten base64-Werte als `int`-Parameter übergeben.

  Bevor die Bitmanipulationsoperatoren auf die Parameter angewandt werden, sollen Sie sicherstellen, dass der Wertebereich zwischen 0 ... 63 liegt.
  Danach wird das resultierende Byte erzeugt und als Funktionswert dem Aufrufer zurückgegeben.

Weiter werden 2 Funktionen zur Ein- bzw. Ausgabe benötigt:

- `void` **`getFilteredString`**`(char block[], unsigned maxSize);`
  Liest von der Standardeingabe (Tastatur) wiederholend Zeichen ein. Sollte das eingelesene Zeichen kein Whitespace-Character - also Leerzeichen (`' '`), ein Tabulator (`'\t'` bzw. `'\v'`) oder ein Zeilenvorschubzeichen (`'\n'` oder `'\r'`) sein – werden die Zeichen nachfolgend im Array `block` abgelegt werden.
  Zum zeichenweise Einlesen von der Tastatur wird die Funktion `int getchar();` der Standardbibliothek (Headerdatei: `stdio.h`) verwendet.
  Das Einlesen wird solange wiederholt bis die Eingabefunktion `EOF` liefert oder `maxSize` Werte in das Array geschrieben wurden.
  Nach dem Einlesevorgang muss garantiert werden, dass es sich bei dem eingelesenen String in `block` um einen ASCIIZ-String handelt.
- `void` **`putData`**`(const char block[], unsigned blockSize);`
  Gibt die Zeichen des Arrays `block` auf der Standardausgabe (Bildschirm) aus. Das char-Array wird hierbei nicht als ASCIIZ-String betrachtet, d. h. die Anzahl der auszugebenden Zeichen wird als Parameter `blockSize` übergeben.
  Die Zeichen des Arrays werden mit der Standardbibliotheksfunktion `int putchar(int ch);` (Headerdatei: `stdio.h`) ausgegeben.

Algorithmen und Datenstrukturen
Versuch 5

VERSUCHSDURCHFÜHRUNG

- Melden Sie sich mit Ihrem Account an und öffnen Sie ein Terminal-Fenster.
- Erstellen Sie ein C-Projekt mithilfe von Eclipse.
  Die Vorgehensweise wird im Dokument
  `http://pc01-lsw.ee.hm.edu/wiki/EclipseProjekterstellung`
  beschrieben.
- Das Arbeitsbereichverzeichnis soll `versuch5` lauten und das Projektverzeichnis `versuch5a`.
- Falls Ihre Umsetzung (mit den Compiler-Schaltern "`-Wall -pedantic -ansi`") Warnungen oder Fehler erzeugt, korrigieren Sie Ihren Quellcode.
- Implementieren Sie die fehlenden Funktionen in der Datei `b64utils.c`

- Korrigieren Sie die fehlerhafte Quellcodedatei `b64decode.c`.

- Testen Sie den Debugger indem Sie einen geeigneten Haltepunkt in der Funktion `b64decode.c` setzen.
- Damit Sie besser Debuggen können sollten Sie zuerst die Zeile
  `getFilteredString(src, BLOCK_SIZE);`
  in `versuch5.c` auskommentieren und den String `src` mit der Stringkonstanten "`V28/`" initialisieren.
  Hinweis: Als dekodierter String muss "`Wo?`" in der Ausgabe erscheinen.

- Machen Sie die Änderungen in `versuch5.c` wieder rückgängig und testen Sie das Programm im Terminal (**außerhalb von Eclipse**) mit dem Inhalt der zur Verfügung gestellten Datei `b64bsp1.txt`.

  ACHTUNG: Das integrierte Konsolenfenster, das Ihnen Eclipse für die Ein-/Ausgabe zur Verfügung stellt ist (insbesondere in der Windowsversion von Eclipse) fehlerhaft!
  Testen Sie deshalb die Funktionen `getFilteredString` und `putData` unbedingt mithilfe eines eigenen Terminalfensters bzw. einer eigenen Eingabeaufforderung.

- Lassen Sie sich das lauffähige Programm und Ihre Struktogramme von Ihrem Betreuer abnehmen.