Dipl.Phys. Gerald Kempfer

Berliner Hochschule für Technik
- University of Applied Sciences -
Fachbereich VI – Informatik und Medien
Studiengang Technische Informatik Bachelor

Lehrveranstaltung "Grundlagen der Informatik"
Übungsblatt 8

Hinweise:

Dieses Übungsblatt ist zur Zulassung zu der Klausur erfolgreich zu bearbeiten ("*Erfolgreich*" bedeutet: Keine Programmabstürze bzw. Endlosschleifen, Aufgabenstellung einschl. der Nebenbedingungen müssen eingehalten sowie Kommentierung und Einrückung korrekt sein! Compilerwarnungen sollen möglichst vermieden werden.).

Die Aufgaben werden überwiegend in den Übungszeiten bearbeitet und dort auch abgegeben. Allerdings genügt die Zeit hierfür unter Umständen nicht, so dass Sie auch außerhalb dieser Zeiten die Aufgaben bearbeiten müssen. Der Abgabetermin für diese Aufgabe ist der **13. Juni 2025**.

Aufgabe: Ziel dieser Übung ist das Verwenden von Bitoperatoren und Funktionen.

Erstellen Sie einen Bitoperatoren-Rechner, d.h. einen Taschenrechner, mit dem vor allem mit Bitoperationen (`&`, `|`, `^`, `<<`, `>>`; wer möchte, kann auch die Grundrechenarten und das Modulo mit einbauen) gerechnet werden kann (siehe Beispiel unten). Dabei sollen die Zahlen (`short`, also nur 16 Bit!) in der Ergebnisdarstellung tabellarisch im dezimalen, oktalen, hexadezimalen und binären Format angezeigt werden. Optisch soll das Programm mittels Rahmen und Linien strukturiert dargestellt werden (siehe auch hierfür die Beispielausgaben!). Hierbei soll die Headerdatei `escapesequenzen.h` aus dem Skript „Grundlagen der Informatik“ Kapitel 6.3 verwendet werden. Durch Verwenden dieser Headerdatei stehen Ihnen verschiedene Makros zum Steuern des Cursors sowie zum Löschen einer Zeile bzw. des ganzen Bildschirms zur Verfügung.

Nachdem eine Berechnung durchgeführt wurde, soll der Benutzer gefragt werden, ob er noch einmal möchte oder nicht.

Um die `main`-Funktion kurz und übersichtlich zu halten und bestimmte, sich wiederholende Funktionalität nicht mehrmals programmieren zu müssen, sollen mehrere Funktionen erstellt werden.

Im folgenden wird der Gedankengang (als Vorschlag) aufgezeichnet, wie ich zu den Funktionen gekommen bin. Dies soll dabei helfen, die Vorgehensweise beim Umsetzen eines Problems in ein Programm besser zu verstehen. Alle Funktionen werden in der main-Funktion innerhalb der `do-while`-Schleife zur wiederholten Durchführung der Berechnung aufgerufen.

Als erstes wird eine Funktion benötigt, die den Bildschirm löscht (Makros CLEAR und HOME aus der `escapesequenzen.h`) und dann die komplette Eingabemaske (Rahmen, Linien sowie feststehende Texte) auf dem Bildschirm ausgibt:

```
void printFrame();
```

Überlegen Sie sich, wie breit die Ergebnis-Spalten für die verschiedenen Zahlenformate sein müssen.

Als nächstes kommt eine Funktion, in der der erste Operand eingelesen wird. Hier muss zuvor noch die alte Eingabezahl (vom vorigen Programmdurchlauf bzw. von einer vorigen, falschen Eingabe) gelöscht werden. Dazu wird der Cursor an die entsprechende Position in der Eingabemaske gesetzt (Makro POSITION) und der entsprechende Bereich gelöscht (d.h. mit Leerzeichen überschrieben). Dann wird der Cursor wieder zurückgesetzt an die Eingabeposition. Nun kann der Benutzer die Zahl eingeben. Dabei muss natürlich wieder auf Fehleingaben geprüft werden. D.h. diese Funktion darf nur bei der Eingabe einer gültigen Zahl verlassen werden. Dies alles soll folgende Funktion leisten, die die eingelesene Zahl als Funktionsergebnis zurückgibt:

```
short getNumber(int Zeile);
```

Durch die Angabe der Bildschirmzeile, in der die Zahl eingelesen werden soll, als Parameter kann diese Funktion auch für den zweiten Operanden verwendet werden; dort ist nämlich die gleiche Funktionalität erforderlich.

Um die eingegebene Zahl in der Eingabemaske richtig darzustellen und um eventuell eingegebene Eingaben nach der Zahl wieder zu beseitigen, wird eine Funktion benötigt, die den Eingabebereich des Operanden wieder löscht und die Zahl erneut an die Eingabeposition schreibt:

```
void printInputNumber(int Zeile, short Zahl);
```

Um auch Eingaben korrekt darzustellen, die über den Rahmen der Eingabemaske hinausgegangen sind, wird am besten die ganze Zeile mittels dem Makro CLEAR_LINE gelöscht und neu geschrieben (Es wird dabei vorausgesetzt, dass die Eingaben niemals über das Zeilenende hinaus gehen). Durch die Angabe der Bildschirmzeile kann diese Funktion auch wieder für den ersten und zweiten Operanden verwendet werden.

Mit dem gleichen Prinzip werden entsprechend zwei Funktionen für die Eingabe des Operators benötigt. Der Operator soll als ein einzelnes Zeichen eingelesen werden. Bei den Operatoren << und >> werden nur das erste Zeichen < und > eingeben und gespeichert. Beim Schreiben des Operators sollen diese beiden Operatoren wieder als zwei Zeichen ausgeschrieben werden.

```
char getOperator();
void printInputOperator(char Operator);
```

Da nur ein Operator eingegeben werden muss, wird hier der Parameter für die Bildschirmzeile nicht benötigt – die Bildschirmzeile steht entsprechend der Eingabemaske fest.

Nun ist die Eingabe erledigt und das Ergebnis kann berechnet werden. Wie der Name der Funktion schon aussagt, soll hier nur die Berechnung durchgeführt werden; es erfolgen keine Bildschirmausgaben! Die beiden Operanden und der Operator (alles gültige Werte, da die Gültigkeit in den obigen Funktionen geprüft wird) werden als Parameter übergeben; das Ergebnis der Berechnung wird als Funktionsergebnis zurückgegeben:

```
short calcResult(short Z1, short Z2, char Op);
```

Jetzt kommen die Ausgabe-Funktionen. Zuerst muss der Operand in den vier verschiedenen Formaten jeweils rechtsbündig ausgegeben werden: erstens dezimal, zweitens oktal (mit führender 0), drittens hexadezimal (mit führendem 0x) und viertens binär:

```
void printResultNumber(int Zeile, short Zahl);
```

Verwenden Sie Formatierungsanweisungen, um die führende 0 (oktal) bzw. 0x (hexadezimal) in der Ausgabe zu erzeugen!
Durch Angabe der Bildschirmzeile kann diese Funktion für die beiden Operanden sowie für das Ergebnis verwendet werden.
Die binäre Ausgabe soll durch eine weitere Funktion ausgeführt werden:

```
void printBinary(int Zeile, short Zahl);
```

Diese Funktion soll direkt von der Funktion `printResultNumber` aufgerufen werden!

Analog zur Ausgabe der Zahlen wird eine Funktion für die Ausgabe des Operators benötigt. Der Operator soll zwischen den beiden Operandenzeilen in jede Spalte geschrieben werden. Hier sollen wieder die beiden Operatoren << und >> ausgeschrieben werden.

```
void printResultOperator(char Operator);
```

Analog zur Eingabe des Operators wird der Operator nur in einer Zeile entsprechend der Bildschirmmaske ausgegeben; daher wird auch hier die Bildschirmzeile als Parameter nicht benötigt.

Auch für die Benutzerfrage, ob eine weitere Berechnung gewünscht wird, soll eine Funktion erstellt werden. Diese Funktion darf erst verlassen werden, wenn der Benutzer wahlweise j oder n (oder die entsprechenden Großbuchstaben) eingegeben hat. Als Funktionsergebnis wird ein Wahrheitswert (j -> wahr und n -> falsch) zurückgegeben.

```
int askAgain();
```

Zum Löschen des Tastaturpuffers (nach jedem `scanf`-Aufruf! Beim Einlesen eines einzelnen Zeichens mit `scanf` darf diese Funktion nur dann aufgerufen werden, wenn das gelesene Zeichen selbst kein `\n` ist!):

```
void clearBuffer();
```

Es sollen keine globalen Variablen verwendet werden; dafür gibt es schließlich die Funktionsparameter!

Das Programm soll benutzerfreundlich sein, d.h. dem Benutzer soll mitgeteilt werden, was er tun soll und was er falsch gemacht hat.

Jede Funktion soll einen Funktionsheader erhalten (siehe Kapitel 5.3 im Skript „Grundlagen der Informatik“). Es sollen zumindest die Informationen Funktionsname, Beschreibung (was macht die Funktion?) sowie Beschreibung der Parameter und des Funktionsergebnisses enthalten sein.

Das Compilieren, Linken und Starten des Programms soll wieder mittels einer Make-Datei durchgeführt werden.

Beispiele: *(Eingaben sind grau hinterlegt)*

```
|-----------------------------------------------------------|
| Bitoperatoren-Rechner                                     |
|                                                           |
| Eingabe Zahl 1: 31                                        |
| Operator:       |                                         |
| Eingabe Zahl 2: 224                                       |
|                                                           |
|-----------------------------------------------------------|
|                                                           |
|          |  dez.  |  okt.   |  hex.  | Binaerdarstellung |
| Zahl 1   |     31 |     037 |   0x1f | 0000000000011111  |
| Operator |      | |       | |      | |                 |  |
| Zahl 2   |    224 |    0340 |   0xe0 | 0000000011100000  |
| -------------------------------------------------------- |
| Ergebnis |    255 |    0377 |   0xff | 0000000011111111  |
|                                                           |
|-----------------------------------------------------------|

Moechten Sie noch einmal (j/n)? Ja


|-----------------------------------------------------------|
| Bitoperatoren-Rechner                                     |
|                                                           |
| Eingabe Zahl 1: 1                                         |
| Operator:       <                                         |
| Eingabe Zahl 2: 15                                        |
|                                                           |
|-----------------------------------------------------------|
|                                                           |
|          |  dez.  |  okt.   |  hex.  | Binaerdarstellung |
| Zahl 1   |      1 |      01 |    0x1 | 0000000000000001  |
| Operator |     << |      << |     << |                <<  |
| Zahl 2   |     15 |     017 |    0xf | 0000000000001111  |
| -------------------------------------------------------- |
| Ergebnis | -32768 | 0100000 | 0x8000 | 1000000000000000  |
|                                                           |
|-----------------------------------------------------------|

Moechten Sie noch einmal (j/n)? ja, vielleicht


|-----------------------------------------------------------|
| Bitoperatoren-Rechner                                     |
|                                                           |
| Eingabe Zahl 1: -32768                                    |
| Operator:       >                                         |
| Eingabe Zahl 2: 15                                        |
|                                                           |
|-----------------------------------------------------------|
|                                                           |
|          |  dez.  |  okt.   |  hex.  | Binaerdarstellung |
| Zahl 1   | -32768 | 0100000 | 0x8000 | 1000000000000000  |
| Operator |     >> |      >> |     >> |                >>  |
| Zahl 2   |     15 |     017 |    0xf | 0000000000001111  |
| -------------------------------------------------------- |
| Ergebnis |     -1 | 0177777 | 0xffff | 1111111111111111  |
|                                                           |
|-----------------------------------------------------------|

Moechten Sie noch einmal (j/n)? J
```

```
|-----------------------------------------------------------|
| Bitoperatoren-Rechner                                     |
|                                                           |
| Eingabe Zahl 1: 12345                                     |
| Operator:       %                                         |
| Eingabe Zahl 2: 1000                                      |
|                                                           |
|-----------------------------------------------------------|
|                                                           |
|          |  dez.  |  okt.   |  hex.  | Binaerdarstellung |
| Zahl 1   |  12345 |  030071 | 0x3039 | 0011000000111001  |
| Operator |      % |       % |      % |                 % |
| Zahl 2   |   1000 |   01750 |  0x3e8 | 0000001111101000  |
| -------------------------------------------------------- |
| Ergebnis |    345 |    0531 |  0x159 | 0000000101011001  |
|                                                           |
|-----------------------------------------------------------|

Moechten Sie noch einmal (j/n)? nö
```