Datentypen; Variable

In einem Computer werden zu verarbeitende Informationen im Speicher durch bestimmte Bitkombinationen in Variablen/Konstanten abgelegt. Jede Variable/Konstante hat dabei eine eigene Speicheradresse und ist von einem bestimmten Datentyp. Dieser Datentyp bestimmt,

- wie eine gegebene Bitkombination interpretiert wird,
- den Wertebereich, den Variablen/Konstanten dieses Typs annehmen können
- die Operationen, die mit Operanden dieses Typs erlaubt sind und wie sie ausgeführt werden.

In diesem Praktikum soll vermittelt werden, dass jede Variable Speicher in bestimmter Länge an einer bestimmten Adresse belegt. Es sollen Techniken zur Eingabe von der Konsole sowie zur formatierten Ausgabe kennen gelernt werden.

Schreiben Sie ein C-Programm, das je eine Variable der eingebauten Datentypen

- signed char, unsigned char
- signed short, unsigned short
- signed int, unsigned int
- signed long, unsigned long

definiert.

Jede dieser Variablen soll über eine Eingabe von der Standardeingabe (Tastatur/Konsole) einen Wert zugewiesen bekommen. Die Eingabe erfolgt zunächst als Zeichenkette in einen Puffer vBuf. Danach erfolgt die Umwandlung in eine Zahl mit Hilfe der Funktionen atoi oder atol. Für Unsigned-Werte stellen diese Funktionen kein Optimum dar, für's Erste soll es aber so hingehen.
Für die Umwandlung von Werten des Typs float oder double kann die Funktion atof benutzt werden.

Die Ausgabe der Werte soll in dezimaler und hexadezimaler Form erfolgen, außerdem soll die Adresse ausgegeben werden. Dies erfolgt mit der Funktion printf.
printf und scanf arbeiten mit sogenannten Formatsteuerzeichen, diese beginnen immer mit einem Prozentzeichen. Aus dem Quelltext können Sie ersehen, für welche Datentypen die zugehörigen Formatsteuerzeichen verwendet werden.

Der Quelltext könnte in etwa folgendermaßen aussehen:

TypesAndVars3.c - KWrite

Datei Bearbeiten Ansicht Lesezeichen Extras Einstellungen Hilfe

```c
/*****************************************************************
Musterloesung Praktikum Programmiersprachen C
Arnold Beck
*****************************************************************/


#include <stdio.h>
#include <stdlib.h>

char vBuf[128];

int main()
{
    signed   char   my_sc;
    unsigned char   my_uc;
    signed short    my_ss;
    unsigned short  my_us;
    signed int      my_si;
    unsigned int    my_ui;
    signed long     my_sl;
    unsigned long   my_ul;
    float           my_f;
    double          my_d;

    fgets(vBuf,128,stdin); my_sc=atoi(vBuf);          /* Eingabe Bytewert */
    fgets(vBuf,128,stdin); my_uc=atoi(vBuf);          /* Eingabe Bytewert */
    fgets(vBuf,128,stdin); my_ss=atoi(vBuf);          /* Eingabe Shortwert */
    fgets(vBuf,128,stdin); my_us=atoi(vBuf);          /* Eingabe Shortwert */
    fgets(vBuf,128,stdin); my_si=atoi(vBuf);          /* Eingabe Intwert */
    fgets(vBuf,128,stdin); my_ui=atoi(vBuf);          /* Eingabe Intwert */
    fgets(vBuf,128,stdin); my_sl=atol(vBuf);          /* Eingabe Longwert */
    fgets(vBuf,128,stdin); my_ul=atol(vBuf);          /* Eingabe Longwert */



    printf("signed   char  %p %d %X\n",&my_sc,my_sc,my_sc);
    printf("unsigned char  %p %d %X\n",&my_uc,my_uc,my_uc);
    printf("signed   short %p %d %X\n",&my_ss,my_ss,my_ss);
    printf("unsigned short %p %d %X\n",&my_us,my_us,my_us);
    printf("signed   int   %p %d %X\n",&my_si,my_si,my_si);
    printf("unsigned int   %p %d %X\n",&my_ui,my_ui,my_ui);
    printf("signed   long  %p %d %X\n",&my_sl,my_sl,my_sl);
    printf("unsigned long  %p %d %X\n",&my_ul,my_ul,my_ul);
  return 0;
}
```

Erstellen Sie ein Arbeitsverzeichnis C_prakt und wechseln Sie in dieses.
Erfassen Sie den Quelltext mit einem Editor Ihrer Wahl (kwrite, kate, vi ...).
Speichern Sie die Datei unter prakt3.c.
Kompilieren Sie "gcc -o prakt3 prakt3.c".
Wenn Sie gut gearbeitet haben, sollten Sie keine Fehlermeldungen erhalten.
Testen Sie Ihr Programm durch Aufruf von "./prakt3".

Fertigen Sie eine Skizze (mit Stift und Papier) an, in der Sie die Speicherplätze mit ihren Adressen und Werten aufzeichnen.

Testen Sie das Programm mit dem Debugger "kdbg". Hierzu müssen Sie Ihr Programm mit der Compileroption -g kompilieren (gcc -g -o prakt3 prakt3.c).

Zusatz: Ändern Sie das Programm in einer neuen Quelltextdatei (prakt3a.c) in nachfolgender Weise. Die Funktion sscanf ist sehr beliebt, weil sie viele Konvertierungen ermöglicht. Aber Vorsicht! dieses Programm birgt verborgene Fehler und funktioniert wahrscheinlich nicht erwartungsgemäß.

```c
#include <stdio.h>
#include <stdlib.h>

char vBuf[128];

int main()
{
    signed   char   my_sc;
    unsigned char   my_uc;
    signed short    my_ss;
    unsigned short  my_us;
    signed int      my_si;
    unsigned int    my_ui;
    signed long     my_sl;
    unsigned long   my_ul;
    float           my_f;
    double          my_d;

    fgets(vBuf,128,stdin); sscanf(vBuf,"%d" ,&my_sc);          /* Eingabe Bytewert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%u" ,&my_uc);          /* Eingabe Bytewert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%hd",&my_ss);          /* Eingabe Shortwert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%hu",&my_us);          /* Eingabe Shortwert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%d" ,&my_si);          /* Eingabe Intwert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%u" ,&my_ui);          /* Eingabe Intwert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%ld",&my_sl);          /* Eingabe Longwert */
    fgets(vBuf,128,stdin); sscanf(vBuf,"%lu",&my_ul);          /* Eingabe Longwert */

    printf("  signed char  %p %d %X\n",&my_sc,my_sc,my_sc);
    printf("unsigned char  %p %u %X\n",&my_uc,my_uc,my_uc);
    printf("  signed short %p %d %X\n",&my_ss,my_ss,my_ss);
    printf("unsigned short %p %u %X\n",&my_us,my_us,my_us);
    printf("  signed int   %p %d %X\n",&my_si,my_si,my_si);
    printf("unsigned int   %p %u %X\n",&my_ui,my_ui,my_ui);
    printf("  signed long  %p %ld %X\n",&my_sl,my_sl,my_sl);
    printf("unsigned long  %p %lu %X\n",&my_ul,my_ul,my_ul);

  return 0;
}
```

Merke:
Die Eingabe ganzahliger Werte ist über
`fgets(vBuf,128,stdin); xyz=atoi(vBuf);`
am sichersten und sollte immer in dieser Weise geschehen.

A. Beck