Hochschule Deggendorf Dr. Peter Jüttner	
Vorlesung: Einführung in die Programmierung	WS 2012
Übung 13	Termin 15.1.13

Pointer (Zeiger)

Bearbeiten Sie die folgenden Aufgaben in Gruppen

Theorie

A1. Theorie

1. Welchen Typ von Werten speichern Pointervariable?

   Pointervariable speichern Adressen

2. Wie kann man sich eine Pointervariable „bildlich“ vorstellen?

   Als Zeiger auf eine Speicherstelle, die in der Pointervariablen gespeichert ist.

   pointervariable 

3. Wie wird eine Pointervariable deklariert? Geben Sie ein Beispiel

   Eine Pointervariable wird deklariert durch eine Typangabe, einen * und den Namen der Variablen. Beispiel für einen Pointer auf int: int *ip;

4. Wie wird ein Wert an eine Pointervariable zugewiesen?

   Wie gewohnt durch den Zuweisungsoperator, Pointervariable = Wert, z.B. ip = 0;

5. Welche Werte werden sinnvollerweise an Pointervariable zugewiesen?

   0 oder NULL (Nullpointer), ip = 0;
   Adresse einer Variablen, ip = &x;
   Konstanter Wert, ip = 0x12345678;

6. Was versteht man unter Dereferenzieren? Wie wird eine Pointervariable dereferenziert

Dereferenzieren = Zugriff auf den Inhalt der Speicherzelle(n), deren Adresse, in der Pointervariablen gespeichert ist.
*-Operator wird vor die Variable gesetzt.

Bildlich:

7. In folgendem Programmstück werden Variable ivar und ivar2 vom Typ Interger deklariert und Pointervariable ip und ip1 vom Typ Pointer auf Integer. Danach werden einige Zuweisungen ausgeführt. Nehmen Sie an die Variable ivar hat die Adresse 0x12345678 und ivar2 hat die Adresser 0x11223344 Welchen Wert haben die Variablen ip und ivar nach jeder Zuweisung? Stellen Sie die Zuweisungen bildlich dar

int *ip = 0;	ip hat den Wert 0	ip	0
int ivar = 10;	ivar hat den Wert 10	ivar	10
ip = &ivar;	ip hat den Wert 0x12345678	ip	0x12345678
*ip = 20;	ivar hat den Wert 20	ivar	20
int ivar2 = 100;	ivar2 hat den Wert 100	ivar2	100
int *ip2 = ip1;	ip2 hat den Wert 0x12345678	ip2	0x12345678
ip1 = &ivar2;	ip hat den Wert 0x11223344	ip	0x11223344
*ip1 = 77;	ivar2 hat den Wert 77	ivar2	77

8. Was bedeutet es, wenn ein Pointer vom Typ void ist?
   Pointer ist „reine“ Adresse, dem Speicher auf den der Pointer zeigt, ist kein Typ zugeorndet
9. Wie kann ein void-Pointer in einen anderen Pointer umgewandelt werden?
   durch Typumwandlung (cast). Der gewünschte Typ (Pointer auf …) wird vor den void-Pointer gesetzt. z.B. (int*) vp = …;

10. Was ist der Heap Speicherbereich?
    Teil des Arbeitsspeichers, der dem Programmierer zur eigenen Verwaltung zur Verfügung steht.
11. Wie wird mit dem Heap gearbeitet? (Speicher anfordern und freigeben)
    Speicher anfordern mittels malloc(Anzahl Bytes). malloc liefert einen typlosen Pointer (void*) zurück. Dieser kann/muss auf den Zielpointer gecastet werden. Die Freigabe des Speichers erfolgt durch die Prozedur free(Pointer), die den Speicher auf dem Heap, auf den Pointer zeigt, wider freigibt.

A2. Praxis

1. Programmieren Sie die Codesequenz aus A1.7. Geben Sie dabei die Werte der Variablen nach jedem Schritt aus.

2. Schreiben Sie ein Prozedur Pointerausgabe mit einem Parameter vom Typ void*. Die Prozedur soll den Wert des Pointers ausgeben und den Inhalt des Speicherbereichs, auf den der Parameter zeigt, jeweils interpretiert als char, short, int unsigned int und float. Casten Sie dazu den Pointer auf den Typ bevor sie ihn dereferenzieren
   Rufen Sie die Prozedur mit den Adressen entsprechender Variablen der angegebenen Typen auf.

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

void pointerausgabe(void* p)
{
    printf("Wert des Pointers:%p\n",p);
    printf("Als char:%c\n",*((char*)p));
    printf("Als short:%hd\n",*((short*)p));
    printf("Als int:%d\n",*((int*)p));
    printf("Als unsigned int:%u\n",*((unsigned int*)p));
    printf("Als float:%f\n\n",*((float*)p));
};

int main(void)
{
   char c = 'c';
   short s = 17;
   int i = 23456789;
   unsigned int ui = 2345678;
   float f = 1.325e4;

   pointerausgabe((void*) &c);
   pointerausgabe((void*) &s);
   pointerausgabe((void*) &i);
   pointerausgabe((void*) &ui);
   pointerausgabe((void*) &f);

   system("PAUSE");
}
```

3. Schreiben Sie eine Prozedur, die über die Adressen dreier Variablen deren Werte tauscht. Dabei soll die erste Variable den Wert der zweiten, die zweite den Wert der dritten und die dritte den Wert der ersten Variablen erhalten.

```
void tausche(int* a, int*b, int* c)
{
int hilf = *a;
```

```
*a = *b;
*b = *c;
*c = hilf;
};
```

4. Schreiben Sie ein C-Programm, das auf dem Heap Speicher für eine Zahl Typ int reserviert. Schreiben Sie mittels scanf(…) einen Wert direkt über die Adresse in den Speicher, geben Sie den Wert mittles printf wieder aus. Danach geben Sie den Speicher wieder frei.

```
…
int *hp = (int*)malloc(sizeof(int));
printf("Bitte eine Zahl eingeben:\n");
scanf("%d", hp);  /* Hier ist kein Adressoperator notwendig, da Pointer */
printf("Zahl im Speicher:%d\n",*hp); /* Hier Dereferenzierung! */
free (hp);
…
```

5. Schreiben Sie ein C-Programm, das auf dem Heap Speicher für eine Zahl Typ int reserviert. Geben Sie die Adresse des reservierten Speichers mittels printf aus. Geben Sie den Speicher mittels free wieder frei.
   Danach geben Sie den Speicher wieder frei. Reservieren Sie erneut Speicher auf dem Heap und geben Sie die Adresse aus.

```
…
int *hp = (int*)malloc(sizeof(int));
printf("Adresse:%p\n",hp);
free(hp);
hp = (int*)malloc(sizeof(int));
printf("Adresse:%p\n",hp);
…
```