Prof. G. Zachmann
Dipl.-Inf. C. Giesemann

TU Clausthal
Institut für Informatik
19. Januar 2006

Wintersemester 2005/2006

Übungen zu Grundlagen der Programmierung in C - Blatt X

Abgabe vom 25.1.2006 bis 31.1.2006 in der angemeldeten Übung

Aufgabe 1 (Scope und Lifetime, *3 Punkte*)

1. Bestimmen Sie in dem folgenden Code jeweils Scope und Lifetime für die Variablen `nummer`, `zahl`, `wert1`, `wert2`, `letzter_wert` und `numx4` .

```
#include <stdlib.h>
int nummer = 1123;

int fkt( int wert1, int wert2 = 0)
{
    static int letzter_wert = -1;

    if ( letzter_wert == -1 )
    {
        int numx4 = nummer * 4;
        return letzter_wert = wert2 * numx4;
    }
    else
    {
        return letzter_wert += wert1;
    }
}

int main( void )
{
    int zahl = 0;
    nummer = 1123;
    fkt( zahl, nummer+4 );
}
```

Aufgabe 2 (Zufallszahlengenerator, *7 Punkte*)

Hinweise:
Pseudo-Zufallszahlen lassen sich mit einem linearen Kongruenzgenerator der folgenden Form erzeugen:

$$X_n = (a \cdot X_{n-1} + b) \, mod \, m$$

Die Parameter a, b und m bestimmen die Periode der Zufallszahlen, nach denen die Folge der Zahlen von vorne beginnt. Die erste Zahl, also der Beginn der Folge, wird durch die Initialisierung mit dem Parameter $X_0 \in \{0, \ldots, m-1\}$ bestimmt, danach wird, solange keine neue Zufallsfolge beginnen soll, immer der vorherige Wert (lokale statische Variable) zum Bestimmen der nächsten Zahl verwendet.

1. Schreiben Sie ein Programm, das eine eigene Funktion namens `zufall` zur Erzeugung von Zufallszahlen mit einem linearen Kongruenzgenerator verwendet. Diese Funktion soll X_0 als Parameter erhalten und den Zufallswert zurückgeben. Der Parameter in der Funktion soll einen geeigneten Default-Wert (unterschiedlich von den möglichen X_0) haben, damit man erkennen kann, wann die Funktion ohne Parameter aufgerufen wurde, um die nächste Zahl zu erhalten. Das Hauptprogramm soll den Initialisierungswert einlesen und den Benutzer fragen, wie viele Zufallszahlen er möchte. Diese sind zu bestimmen und auszugeben. Wählen Sie die konstanten Parameter der Formel wie folgt: a = 106, b = 1283 und m = 6075.
2. Mit den gewählten Konstanten hat der Generator eine Periode von 6075. Überprüfen Sie dieses, in dem Sie sich insgesamt 6100 einander folgende Zufallszahlen berechnen. Damit die Ausgabe nicht zu unübersichtlich wird, geben Sie nur die ersten und die letzten 25 Zahlen der Zufallsfolge formatiert (siehe Beispiel) aus und vergleichen Sie diese. Beispiel für die Initialisierung mit $X_0 = 3$:

```
    1: 1601
    2:  889
... weitere 22 Zeilen
   25: 4628
6076: 1601
6077:  889
... weitere 22 Zeilen
6100: 4628
```

Aufgabe 3 (Function Overloading, *4 Punkte*)

1. Schreiben Sie ein Programm, das 4 Funktionsprototypen enthält, die alle den Namen `increment` haben, sich aber im Datentyp des Parameters unterscheiden. Jede Funktion soll einen übergebenen Parameter um 1 erhöhen und zurückgeben. Die 4 unterschiedlichen Datentypen der Parameter sollen wie folgt sein: `int`, `float`, `char` und `farbe`. Dabei soll `farbe` als `enum` mit rot, gelb, gruen, blau, weiss und schwarz deklariert werden. Fordern Sie im Hauptprogramm den Benutzer nacheinander auf, einen Wert des jeweiligen Typs einzugeben und geben Sie jeweils das um 1 erhöhte Ergebnis aus.

Aufgabe 4 (Pointer auf Structs, *6 Punkte*)

Hinweise:
Aufgrund der gegenseitigen Abhängigkeit in den Definitionen der Structs Person und Abteilung benötigen Sie eine *Forward Declaration*[1].

Gegeben seien die folgenden Definitionen der Structs Person und Abteilung:

```
struct Person
{
    int personalNr;
    Abteilung* abteilung;
};
```

```
struct Abteilung
{
    int abteilungsNr;
    Person* leiter;
};
```

[1] Zur Erinnerung: Im allgemeinen müssen Structs in C++ *vor* ihrer Benutzung *definiert* worden sein. Um innerhalb einer Variablen- bzw. Member-Definition einen Pointer auf eine Struct verwenden zu können (wie in obigem Beispiel), genügt es aber auch, diesen Struct vorher zu *deklarieren* (*Forward Declaration*). Analog zur Funktions-Deklaration gibt die *Forward Declaration* einer Struct nur den Namen der Struct an, nicht aber deren Inhalt (Member). Die Syntax der *Forward Declaration* lautet für eine Struct namens S wie folgt (man beachte die fehlenden geschweiften Klammern):
`struct S;`

1. Schreiben Sie eine Funktion, welche eine Person (oder alternativ einen Pointer auf eine Person) als Parameter erhält und für diese Person folgende Daten ausgibt:
   - ihre Personalnummer
   - die Nummer der Abteilung, bei der die Person angestellt ist
   - ob die Person Leiter dieser Abteilung ist (Hinweis: Pointer-Vergleich)
   - falls nicht, die Personalnummer des zugehörigen Abteilungsleiters
2. Schreiben Sie zum Testen dieser Funktion ein Hauptprogramm, welches die folgenden Variablenbelegungen verwendet und die obige Funktion für alle fünf Personen aufruft:

```
Person anton, berta, caesar, dora, emil;
Abteilung lager, verkauf;

anton.personalNr  = 105;  anton.abteilung  = &lager;
berta.personalNr  = 114;  berta.abteilung  = &lager;
caesar.personalNr = 123;  caesar.abteilung = &lager;
dora.personalNr   = 132;  dora.abteilung   = &verkauf;
emil.personalNr   = 141;  emil.abteilung   = &verkauf;

lager.abteilungsNr   = 1;  lager.leiter   = &berta;
verkauf.abteilungsNr = 2;  verkauf.leiter = &dora;
```