Winterterm 2012/2013

Otto-von-Guericke-University Magdeburg
Max Planck Institute for Dynamics of Complex Technical Systems
Computational Methods for Systems and Control Theory

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2012_WS_SC/

Scientific Computing 1
Tutorial 1a
10/18/2012

Solution

Exercise 1:

Choose an editor and write a program which prints your name to the screen. Save it as `myname.c` and compile it to an executable binary called `myname`. Installed editors inside the virtual machine are:

- `gedit`
- `nedit`
- `nano`
- `vim` and `gvim`
- `emacs`

All editor commands can be followed by the name of a file that is opened if it exists or created if it does not exist.

Solution:

```c
#include <stdio.h>

int main ( int argc, char ** argv ) {
  printf("Max␣Muster\n");
  return 0;
}
```

Exercise 2:

Write a C program which reads a date as an integer number of the format `DDMMYYYY` from the standard input. It should then identify the three parts `DD`, `MM` and `YYYY` and prints them as a normal date to the screen. A check if the date is valid is not necessary.

Solution:

```c
// Standard Header einbinden
#include <stdio.h>
#include <stdlib.h>

// Hauptprogramm
int main(int argc, char ** argv) {
        int TT, MM, JJJJ, eingabe;
        printf("Geben Sie das Datum in der Form TTMMJJJJ ein:");
        scanf("%d",&eingabe);
        JJJJ = eingabe % 10000;
        eingabe = eingabe / 10000;
        MM = eingabe %100;
        TT = eingabe / 100;
        printf("Datum: %d.%d.%d\n",TT,MM,JJJJ);
        return 0;
}
```

Ausgabe:

```
$ gcc datum_modulo.c
$ ./a.out
Geben Sie das Datum in der Form TTMMJJJJ ein:10071986
Datum: 10.7.1986
```

Exercise 3:

Write a C program which reads two integers a and b and computes $\frac{a}{b}$.

a.) Find out what happens if b is zero.

b.) Does the compiler recognize if there is a hard-coded division by zero?

c.) Rewrite the program to floating point numbers. What happens now if b is equal to zero?

d.) Modify the program such that $b = 0$ is detected and avoided before an error occurs.

Solution:

```c
#include <stdio.h>
int main (int argc, char ** argv){
  int a, b;
  scanf("%d", &a);
  scanf("%d", &b);
  printf("a/b = %g\n",a/b);
  return 0;
}
```

a.) `Floating point exception (core dumped)`

b.) `warning:  division by zero [-Wdiv-by-zero]`

c.) Results in `inf`

d.)

```c
#include <stdio.h>
int main (int argc, char ** argv){
  int a, b;
  scanf("%d", &a);
  scanf("%d", &b);
  if ( b != 0 ) {
    printf("a/b = %g\n",a/b);
  } else {
    printf("Error div by zero\n");
  }
  return 0;
}
```

Exercise 4:

Write a C program which reads a floating point number from the standard input and rounds it correctly to the next integer. Hint: A typecast to `int` simply truncates the decimal places.

Solution:

```
#include <stdio.h>
int main ( int argc, char ** argv ) {
  double x;
  int r;
  scanf("%lg\n", &x);
  r = (int)(x+0.5);
  printf("round(%lg)␣=␣%d\n",x, r);
  return 0;
}
```

Exercise 5:

Write a C program which reads two integers a and b from the standard input.

a.) Print the square of intergers from $[a, b)$ on the screen.

b.) What needs to be modified to include b in the set?

c.) Neglect all squares that are odd.

Solution:

a.)
```
#include <stdio.h>
int main ( int argc, char ** argv ) {
  int a, b, i;
  scanf("%d\n", &a);
  scanf("%d\n", &b);
  for (i = a; i < b ; i++) {
    printf("␣%d␣*␣%d␣=␣\n", i,i,i*i);
  }
  return 0;
}
```

b.) Change < to <=:

```
#include <stdio.h>
int main ( int argc, char ** argv ) {
  int a, b, i;
  scanf("%d\n", &a);
  scanf("%d\n", &b);
  for (i = a; i < b ; i++) {
    printf("␣%d␣*␣%d␣=␣\n", i,i,i*i);
  }
  return 0;
}
```

```
#include <stdio.h>
int main ( int argc, char ** argv ) {
  int a, b, i;
  scanf("%d\n", &a);
  scanf("%d\n", &b);
  for (i = a; i < b ; i++) {
    if ( i%2 == 0 )
     printf("␣%d␣*␣%d␣=␣\n", i,i,i*i);
  }
  return 0;
}
```

Exercise 6:

Write a C program which reads an integer value from the standard input and reverses the order of the digits. For example, if the user inputs `4711` the output must be `1174`. The intermediate results should be stored in an integer too.

Solution:

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

int main ( int argc, char ** argv ) {
        int in, out;
        printf("Input:␣");
        scanf("%d",&in);
        out = 0;
        while ( in != 0 ) {
                out = out * 10;
                out = out + (in % 10 ) ;
                in = in / 10;
        }
        printf("Output:␣%d\n",out);
        return 0;
}
```

Exercise 7:

Consider the following program to check if an integer is prime or not:

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

int main (int argc, char ** argv ) {
        int in, i;
        int is_prime;
        printf("Number to check: ")
        scanf("%d",&in);
        if ( in = 2) {
                printf("Two is the oddest prime ;-).\n");
        } else {
                is_prime = 1;
                for (i=3; i < in; i=+2){
                        if (in % I == 0) is_prime=0;
                }
                if ( is_prime )
                        printf("%d is prime.\n,in);
                else
19                              printf("%d is not prime.\n",in);
20          }
21          return 0;
22  }
```

Download it from: http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2012_WS_SC/data/prime.c and try to compile it. Find all errors and fix them.

Solution:

c.) Missing `;` after printf in line 7

- `if ( in = 2)` in line 9
- `=+` instead of `+=` in line 13
- Uppercase `i` in line 14
- Missing " in line 17