Übung zur Vorlesung Informatik I WS 2024/25
Fakultät für Angewandte Informatik
Institut für Informatik
Prof. Dr. J. Hähner, J. Linne, H. Cui, V. Gerling, N. Kemper

Übungsgruppe 69

Abgabe des 4. Übungsblatts

Erik Wiedmann, Marwin Merkl, Manuel Henker

Aufgabe 25+26

Datei: `2526.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
#include <string.h>

#define CODE_LENGTH 4
#define MAX_ATTEMPTS 10

#define VALID_INPUT 0
#define INPUT_CHAR 1
#define INVALID_LENGTH 2

void generate_code(int code[]);
int read_guess(char guess[]);
void flush();
int evaluate_guess(const int code[], const int guess[], int result[]);

int main(void) {
    int code[CODE_LENGTH];
    char input_str[CODE_LENGTH + 5];
    int input[CODE_LENGTH];
    int result[2];
    int try, input_valid, i;

    srand(time(NULL));

    generate_code(code);

    printf("Errate den geheimen Code\n");
    printf("Sie haben 10 Versuche\n");
    printf("-------------------------\n\n");

    for(try = 0; try < MAX_ATTEMPTS; try++) {
        printf("Dein %d. Versuch:\nBitte geben Sie eine 4 stellige Zahl ein:\n\n", (try + 1));
        input_valid = read_guess(input_str);
```

```
37
38          if(input_valid == INPUT_CHAR) {
39              printf("Ungueltige Eingabe! Bitte nur Zahlen eingeben\
                    n");
40              continue;
41          } else if(input_valid == INVALID_LENGTH) {
42              printf("Ungueltige Eingabe! Bitte geben sie genau 4
                    Zahlen ein.\n");
43              continue;
44          }
45
46          for(i = 0; i < CODE_LENGTH; i++) {
47              input[i] = input_str[i] - '0';
48          }
49
50          if(evaluate_guess(code, input, result)) {
51              printf("Herzlichen Glueckwunsch!! Sie haben den Code
                    erraten\n");
52              break;
53          } else {
54              printf("Anzahl der korrekten Zahlen an der richtigen
                    Position: %d\n", result[0]);
55              printf("Anzahl der korrekten Zahlen an der falschen
                    Position %d\n", result[1]);
56          }
57      }
58
59      if(try == MAX_ATTEMPTS) {
60          printf("\nLeider haben sie den Code nicht erraten. Der
                Code war: ");
61          for(i = 0; i < CODE_LENGTH; i++) {
62              printf("%d", code[i]);
63          }
64      }
65
66      printf("\nDanke fürs spielen!\n");
67      return 0;
68  }
69
70  void flush() {
71      int c;
72      while((c = getchar()) != '\n' && c != EOF);
73  }
74
75  void generate_code(int code[]) {
76      int i;
77      for(i = 0; i < CODE_LENGTH; i++) {
78          code[i] = rand() % 10;
79      }
80  }
81
82  int read_guess(char guess[]) {
83      int i;
84      char c;
85
86      while((c = getchar()) != '\n' && c != EOF) {
```

```
87          if(!isdigit(c)) {
88              flush();
89              return INPUT_CHAR;
90          }
91          guess[i] = c;
92          i ++;
93      }
94      if(strlen(guess) != 4) {
95          flush();
96          return INVALID_LENGTH;
97      }
98      return VALID_INPUT;
99  }
100
101 int evaluate_guess(const int code[], const int guess[], int
        results[]) {
102         int i;
103     int code_count[10] = {0};
104         int guess_count[10] = {0};
105
106         for (i = 0; i < CODE_LENGTH; i++) {
107             if (code[i] == guess[i]) {
108                     results[0]++;
109             } else {
110                 code_count[code[i]]++;
111                     guess_count[guess[i]]++;
112             }
113         }
114
115         for (i = 0; i < 10; i++) {
116             results[1] += (code_count[i] < guess_count[i] ?
                    code_count[i] : guess_count[i]);
117         }
118     return results[0] == CODE_LENGTH;
119 }
```

Aufgabe 27

a) Unsicher weil:

- die Zeichenkette zu lang sein kann und der rest verbleibt dann im Puffer
- sollte nichts eingegeben werden, dann blockiert das Programm bis etwas eingegeben wird
- man den Rückgabewert nicht einsehen kann, da dieser nirgends gespeichert wird
- c kann je nach Compilerverhalten überschireben werden, da diese direkt hintereinander gespeichert werde

BSP Eingabe: abcdefghi

b) c) d) Datei: `27bcd.c`

```
1  #include<stdio.h>
2  #include<string.h>
3
4  #define MAX_STRING 10
5  #define ERFOLG 1
```

```c
#define NICHT_ERFOLG 0

void flush(){
    while(getchar() != '\n'){}
}

int read_string(char in[]){
        int status;

        status = scanf(" %9[^\n]", in);
        if(status == 0 || getchar() != '\n'){
                flush();
                return NICHT_ERFOLG;
        }
        return ERFOLG;
}

int count_words(char s[]){
        int i = 0;
        int words = 0;

        if(strlen(s) == 0) return 0;
        while(s[i] != '\0'){
                if(s[i] == ' '){
                        words++;
                }
                i++;
        }
        return words + 1;
}



int main(void){
        char in[MAX_STRING];
        int e;

        printf("\nMaster-sama gib mir pwweeees eine
            Zeichenkette\n mit maximal %i Zeichen, damit ich
            Worterzahlen uben kann m(_ _)m: ", MAX_STRING);

        e = read_string(in);

        if(e == NICHT_ERFOLG){
                printf("\nBAKAAA-sama du hast mir zu viele
                    Zeichen ubergeben grrrrrr");
        }
        else printf("\nDu hast %i Worter geschrieben,
            CHUUUUUUU", count_words(in));

        return 0;
}
```

Aufgabe 28

Datei: `Aufg28.c`

```c
#include <math.h>
#include <stdio.h>
#define DIM 3
#define x 30 /*so viele Stellen kann die Dezimalzahl innerhalb des
    Vektors haben*/

int read_vector(double p[]);
void add_vectors(double p1[], double p2[]);
double vector_length(double p[]);
void own_flush();

int main() {
    double p1[DIM], p2[DIM];
    int i = 0;

    if (read_vector(p1) != 0) {
        return 1;
    }
    if (read_vector(p2) != 0) {
        return 1;
    }

    while(i < DIM) {
        printf("%f ", p1[i]);
        i++;
    }
    printf("\n");
    i = 0;
    while(i < DIM) {
        printf("%f ", p2[i]);
        i++;
    }
    printf("\n");
    i = 0;

    add_vectors(p1, p2);

    while(i < DIM) {
        printf("%f ", p1[i]);
        i++;
    }
    printf("\n");

    printf("%f", vector_length(p1));

    return 0;
}

int read_vector(double p[]) {
    /*+1 für die ';' dazwischen*/
    char input[DIM * x + 1];
    int i = 0, n = 0, temp = 0;

```

```
53      printf("Type three numbers separated by ';': \n");
54      scanf("%s", input);
55
56      while(input[i] != '\0') {
57          if (n > DIM - 1) {
58              printf("Warum gibst du mehr als drei Werte ein, du
                    Dulli \n");
59              own_flush();
60              return 1;
61          } else if (input[i] == ';') {
62              p[n++] = temp;
63              temp = 0;
64          } else {
65              temp = (temp * 10) + input[i] - '0';
66          }
67
68          i++;
69      }
70      /* weil man mit der while-Bedingung den letzen value nicht
           mehr einließt*/
71      p[n] = temp;
72
73      own_flush()
74      if (n != DIM - 1) {
75          printf("Warum gibst du weniger als drei Werte ein, du
                Dulli \n");
76          return 1;
77      }
78      return 0;
79  }
80
81  void add_vectors(double p1[], double p2[]) {
82      int i = 0;
83
84      while(i < DIM) {
85          p1[i] = p1[i] + p2[i];
86          i++;
87      }
88  }
89
90  double vector_length(double p[]) {
91      double underoot = 0;
92      int i = 0;
93
94      while (i < DIM - 1) {
95          underoot += (p[i] * p[i]);
96          i++;
97      }
98
99      return sqrt(underoot);
100 }
101
102 void own_flush() {
103     int c;
104     while ((c = getchar()) != '\n' && c != EOF) {}
105 }
```