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 37

Datei: `37.c`

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

double *max_value(double *v, int v_len){
    int i;
    double *mv = v;
    if(v == NULL) return NULL;
    for(i = 1; i < v_len; i++){
        if(*mv < v[i]) mv = &v[i];
    }

    return mv;
}

double *min_value(double *v, int v_len){
    int i;
    double *minV = v;
    if(v == NULL) return NULL;
    for(i = 1; i < v_len; i++){
        if(*minV > v[i]) minV = &v[i];
    }
    return minV;
}

int normalize_array(double *v, int v_len){
    double *min = min_value(v, v_len);
    double *max = max_value(v, v_len);
    double mm = *max - *min;
    int i;

    if(v == NULL) return -1;
    if(*max - *min <= DBL_EPSILON){
        for(i = 0; i < v_len; i++){
        v[i] = 1;
        }
    }

    for(i = 0; i < v_len; i++){
```

```
        v[i] = v[i] - *min;
        v[i] = v[i] / mm;
    }
    return 1;
}

void print_array(double *v, int v_len){
    int i;
    printf("\n");
    for(i = 0; i < v_len; i++){
        printf("%f ", v[i]);
    }
}

int main(void){
    double deez[5][5] = {{0.0, 1.0, 2.0, 3.0, 4.0}, {3.0, 2.5,
       2.2, 2.7, 2.0}, {1.2, 4.5, 3.7, -0.5, 4.2}, {2.1352,
       -1.0541, 4.5423, 0.231, -3.2441}, {1.0, 1.0 + 0.25 *
       DBL_EPSILON, 1.0 + 0.5 * DBL_EPSILON, 1.0 + 0.75 *
       DBL_EPSILON, 1.0 + DBL_EPSILON}};
    int len = 5;
    print_array(deez[0], len);
    normalize_array(deez[0], len);
    print_array(deez[0], len);
    printf("\n");
    print_array(deez[1], len);
    normalize_array(deez[1], len);
    print_array(deez[1], len);
    printf("\n");
    print_array(deez[2], len);
    normalize_array(deez[2], len);
    print_array(deez[2], len);
    printf("\n");
    print_array(deez[3], len);
    normalize_array(deez[3], len);
    print_array(deez[3], len);
    printf("\n");
    print_array(deez[4], len);
    normalize_array(deez[4], len);
    print_array(deez[4], len);
    return 0;
}
```

Aufgabe 38

Datei: 38.c

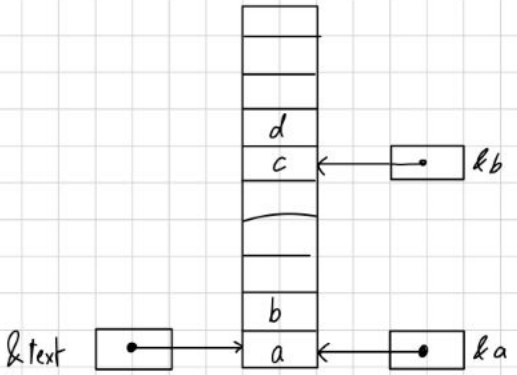
```
#include<stdio.h>
#include<string.h>

int uThere(char x, const char *p){
        int i;
        for(i = 0; i <= (int)strlen(p); i++){
                if(x == p[i]) return 1;          /*Char ist im
                   String vorhanden*/
        }
```

```
9           return 0;                                   /*Char ist nicht
                im String*/
10  }
11
12  int find_longest_substring(const char *source, const char *
        allowed_characters, const char **start){
13          int i;
14          int t = 0, len = 0, c = 0;
15          const char *h = source;
16
17          if(start == NULL) return -1;
18
19          for(i = 0; i < (int) strlen(source); i++){
20                  if(c == t){                                 /*wenn t 2
                        mal das selbe ist gab es eine unterbrechung im
                        substring*/
21                          t = 0;
22                          h = &source[i];
23                  }
24                  c = t;
25                  if(uThere(source[i], allowed_characters)) t++;
26                  if(t > len){
27                          len = t;
28                          *start = h;
29                  }
30          }
31
32          return len;
33  }
34
35  void printN(const char *p, int n){
36          int i;
37          printf("\n");
38          for(i = 0; i < n; i++){
39                  if(p[i] == '\0') break;
40                  printf("%c", p[i]);
41          }
42          printf("\n");
43  }
44
45  int main(void){
46          const char *start = "";
47
48          printN(start, find_longest_substring("Das12ist1235alles1",
                "123456798",&start));
49          return 0;
50  }
```

Aufgabe 39

a) *(5 Minuten)* Zeichen Sie eine Speicherskizze zum betroffenen Bereich des Arbeitsspeichers nach der erfolgreichen Abarbeitung des folgenden Codes:

```
char *text, *a, *b;
text = malloc(10 * sizeof(char));
a = strcpy(text, "ab");
b = strcpy(text + 5, "cd");
```

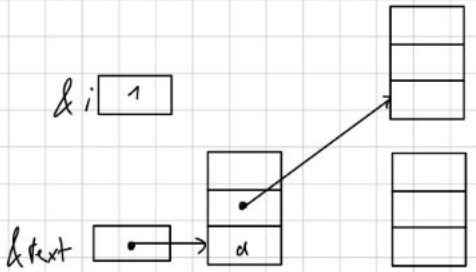


b) *(5 Minuten)* Zeichen Sie eine Speicherskizze zum betroffenen Bereich des Arbeitsspeichers nach der erfolgreichen Abarbeitung des folgenden Codes:

```
char text[3][3], *a, *b;
a = strcpy(text[0], "ab");
b = strcpy(text[2], "cd");
```

c) *(5 Minuten)* Zeichen Sie eine Speicherskizze zum betroffenen Bereich des Arbeitsspeichers nach der erfolgreichen Abarbeitung des folgenden Codes:

```
int i;
char **text = malloc(3 * sizeof(char*));
for (i = 0; i < 2; i++) {
        text[i] = malloc(3 * sizeof(char));
}
strcpy(text[0], "a");
```



d) *(3 Minuten)* Zeichen Sie eine Speicherskizze zum betroffenen Bereich des Arbeitsspeichers nach der erfolgreichen Abarbeitung des folgenden Codes:

```
int i;
char **text = malloc(3 * sizeof(char*));
for (i = 0; i < 3; i++) {
        text[i] = malloc(3 * sizeof(char));
}
for (i = 0; i < 3; i++) {
        free(text[i]);
}
text[2] = NULL;
```