Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WiSe 2024/2025
Fakultät für Angewandte Informatik
Institut für Informatik
Prof. Dr. J. Hähner, J. Linne, H. Cui, V. Gerling, N. Kemper 22.11.2024

Übungsblatt 7

Abgabe: bis 02.12.2024 10:00 Uhr

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt!).
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.

* leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe

Hinweis:
Auf diesem Übungsblatt befinden sich ausschließlich Programmier-Aufgaben. Für **jede** der Programmieraufgaben (auch auf zukünftigen Blättern) gilt: Ihre Programme sollen **kompilierbar** und **ausführbar** sein. Beim Kompilieren Ihrer Programme mit den Compiler-Schaltern `-ansi -pedantic -Wall -Wextra` dürfen **keine** Warnungen oder Fehlermeldungen auftauchen. Lösungen, die sich nicht an diese Vorgaben halten, können Punktabzüge erhalten.

Aufgabe 21 * *(C-Ausdrücke & C-Kontrollstrukturen, 22 Minuten)*

a) *(*, Auswertung von Ausdrücken, 7 Minuten)* Tragen Sie in der folgenden Tabelle jeweils den Wert des gegebenen Ausdrucks ein:

Ausdruck	Wert des Ausdrucks
`!0`	0
`!'0'`	0
`(5 > 3) && (2 <= 2)`	1
`'A' + 1 == 'B'`	1
`'a' != 0x61`	0
`3 ? (2 ? -1 : 0) : 1`	-1
`x = y = 3`	1

Gehen Sie im Folgenden davon aus, dass `x` eine `int`-Variable ist, die vor Beginn der Auswertung des jeweiligen Ausdrucks immer den Wert 2 hat. Tragen Sie in die jeweilige Spalte den Wert des Ausdrucks bzw. den Wert von `x` nach der Auswertung des Ausdrucks ein.

Ausdruck	Wert des Ausdrucks	Wert von `x` nach Auswertung des Ausdrucks
`x += 3`	5	5
`x == 2`	1	2
`x--`	2	1
`++x`	3	3
`x -= 4`	-2	-2
`x *= -1`	-2	-2
`x = y = 0`	1	0

Blacky010 Blatt 7 außer letzte Aufg — e7764af · 5 days ago

41 lines (34 loc) · 518 Bytes

Code | Blame — Raw

```
Aufgabe 2 1 b)
//Aufgabe b:
//1.
double a;
floor(a);

//2.
(x & 7) == 0

//3.
(a >= b && a <= c) || (a >= c && a <= b)

//Aufgabe c:
//1.
if( i == 4) {
        x = pow(x,2);
} else if( i == 8) {
        x = pow(x,4);
} else return x;

//2.
int count = 0;
while (count < 5) {
        printf("Count in %d\n", count + 1);
        count++;
}

//3.
int i = -1;
int sum = 0;
while (i < n) {
        sum += i * i;
        i++;
}

//4.
for(j = 0; j < size; j++) {
        if((arr[j] % 2) != 0) {
                total += arr[j];
        }
}
```

Blacky010 Blatt 7 finished 2d21d67 · 4 days ago

38 lines (33 loc) · 1.47 KB

Code Blame Aufgabe 22 a) Raw

```c
#include <stdio.h>

int main(void) {

        char str1[] = "info";
        char str2[] = "info";
        char str3[] = "informatik";
        char str4[] = "infm";
        char str5[] = "";

        printf("Vergleiche von:");
        //identisch
        printf("'%s' und '%s': %d\n", str1, str2, string_compare(str1, str2)); // 0
        //unterschiedliche Länge
        printf("'%s' und '%s': %d\n", str1, str3, string_compare(str1, str3)); // < 0
        printf("'%s' und '%s': %d\n", str3, str1, string_compare(str3, str1)); // > 0
        //gleiche Länge aber unterschiedliches Zeichen am Ende
        printf("'%s' und '%s': %d\n", str1, str4, string_compare(str1, str4)); // < 0
        printf("'%s' und '%s': %d\n", str4, str1, string_compare(str4, str1)); // > 0
        //mit Leerer Zeichenkette
        printf("'%s' und '%s': %d\n", str5, str1, string_compare(str5, str1)); // < 0
        printf("'%s' und '%s': %d\n", str5, str5, string_compare(str5, str5)); // 0

        return 0;
}

int string_compare(char v[], char w[]) {
        int i;
        while((v[i] != '\0') && (w[i]) != '\0') {
                if(v[i] != w[i]) {
                        return v[i] - w[i];     //genaue Funktionsweise wie strcmp
                                                //Rückgabewert ist differenz der ASCII-Codes der
                                                //vergleichten Zeichen bei gleichlangen Arrays
                }
                i++;
        }
        return v[i] - w[i];
}
```

```c
#include <stdio.h>

int main(void) {

        char str1[] = "Informatik";
        char str2[] = "Info";
        char str3[] = "Inform";
        char str4[] = "Informatik";
        char str5[] = "";
        char str6[] = "Informatiker";
        char str7[] = "In";
        //kurzes Präfix
        printf("'%s' ist Praefix von '%s': %d\n", str2, str1, string_prefix(str1, str2)); // 1
        //langes Präfix
        printf("'%s' ist Praefix von '%s': %d\n", str3, str1, string_prefix(str1, str3)); // 1
        //gleichlanges Präfix
        printf("'%s' ist Praefix von '%s': %d\n", str4, str1, string_prefix(str1, str4)); // 1
        //leere Zeichenkette als Präfix
        printf("'%s' ist Praefix von '%s': %d\n", str5, str1, string_prefix(str1, str5)); // 1
        //v kürzer als Präfix
        printf("'%s' ist Praefix von '%s': %d\n", str6, str1, string_prefix(str1, str6)); // 0

        return 0;
}

int string_prefix(char v[], char w[]) {
        int i;

        while(w[i] != '\0') {
                if ((v[i] == '\0') || (w[i] != v[i])) return 0;
                i++;
        }

        return 1;
}
```

Code | Blame | 57 lines (45 loc) · 1.53 KB

Aufgabe 23

Raw

```c
#include <stdio.h>

#define VALID 0
#define INVALID 1

int check_lower(const char str[]);
void count_letters(const char str[], int letters[26]);

int main(int argc, char *argv[]) {
        int letters[26] = {0};
        int i;


        if (argc < 2) {
                printf("Das Programm zaehlt die Vorkommen der Kleinbuchstaben in den angegebenen Parameterm.\n");
                printf("Fehler: keine Parameter wurden angegeben. BITTE geben sie Parameter mit Kleinbuchstaben an!");
                return 0;
        }

        for (int i = 1; i < argc; i++) {
                if (check_lowercase(argv[i]) == INVALID) {
                        printf("Fehler: Der Parameter '%s' enthaelt Zeichen, die keine Kleinbuchstaben sind.\n", argv[i]);
                        return 1;
                }
                count_letters(argv[i], letters);
        }

        printf("Buchstabenhäufigkeit:\n");
        for (int i = 0; i < 26; i++) {
                if (letters[i] > 0) {
                        printf("%c: %d\n", 'a' + i, letters[i]);
                }
        }
        return 0;
}
```

```c
        }

        printf("Buchstabenhäufigkeit:\n");
        for (int i = 0; i < 26; i++) {
                if (letters[i] > 0) {
                        printf("%c: %d\n", 'a' + i, letters[i]);
                }
        }
        return 0;
}

int check_lower(const char str[]) {
        int i = 0;

        while (str[i] != '\0') {
                if (str[i] < 'a' || str[i] > 'z') return INVALID;
                i++;
        }

        return VALID;
}

void count_letters(const char str[], int letters[26]) {
        int i = 0;

        while (str[i] != '\0') {
                if (str[i] >= 'a' && str[i] <= 'z') {
                        letters[str[i] - 'a']++;
                }
                i++;
        }
}
```

Code | Blame | 63 lines (48 loc) · 2.26 KB

Aufgabe 24

Raw

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

#define MAX_LENGTH 10

void caesar_encrypt(char text[], int shift, int n_chars);

int main(int argc, char *argv[]) {

        if(argc < 2) {
                printf("Fehler: Es muss mindestens ein Eingabeparameter eingegeben werden!");
                return 1;
        }

        char combined_text[MAX_LENGTH + 1];
        int shift, i;
        int current_length, word_length, space_needed, remaining_space;

        srand(time(NULL));
        shift = (rand() % 25) +1 ;
        printf("der zufaellige Shift für die Verschlüsselung ist: %d\n", shift);

        for (int i = 1; i < argc; i++) {
                word_length = strlen(argv[i]);
                space_needed = current_length > 0 ? 1 : 0;

                if (current_length + space_needed + word_length > MAX_LENGTH) {         //Fall wenn naechstes Wort zu lang ist
                        remaining_space = MAX_LENGTH - current_length - space_needed;   //verbleibende Platz wird ermittelt
                        //wenn nur noch 1 Platz frei ist wird wegen dem '- space_needed' die if false und es wird auch ein Leerze
                        if (remaining_space > 0) {
                                strncat(combined_text, " ", 1);
                                strncat(combined_text, argv[i], remaining_space);
                        }
                        break;
```

```c
                if (current_length + space_needed + word_length > MAX_LENGTH) {         //Fall wenn naechstes Wort zu lang ist
                        remaining_space = MAX_LENGTH - current_length - space_needed;   //verbleibende Platz wird ermittelt
                        //wenn nur noch 1 Platz frei ist wird wegen dem '- space_needed' die if false und es wird auch ein Leerzeicher
                        if (remaining_space > 0) {
                                strncat(combined_text, " ", 1);
                                strncat(combined_text, argv[i], remaining_space);
                        }
                        break;
                }

                if (space_needed == 1) {                //Leerzeichen hinzufuegen, wenn es nicht das erste Wort ist
                        strcat(combined_text, " ");
                }
                strcat(combined_text, argv[i]);
                current_length = strlen(combined_text); //aktuelle Laenge updaten
        }

        caesar_encrypt(combined_text, shift, MAX_LENGTH);

        printf("Verschlüsselte Zeichenkette: %s\n", combined_text);

        return 0;
}

void caesar_encrypt(char text[], int shift, int n_chars) {

        int i;
        for(i = 0; (i < n_chars) || (text != '\0'); i++) {
                if(isalpha(text[i])) {
                        char base = islower(text[i]) ? 'a' : 'A';
                        text[i] = (text[i] - base + shift) % 26 + base;
                }
        }

```