Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WiSe 2022/2023
Fakultät für Angewandte Informatik
Institut für Informatik
Dr. M. Frieb, Dr. J. Mische, V. Schmid, L. Petrak, P. Schalk 16.12.2022

Übungsblatt 9

Abgabe: 09.01.2023 9:00 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

Betreuung während der vorlesungsfreien Zeit über Weihnachten:
Während der vorlesungsfreien Zeit vom 24.12.222 bis zum 6.1.23 finden keine synchronen Betreuungsangebote statt. Bei Fragen erstellen Sie ein Thema im Forum der Informatik 1 Digicampus-Veranstaltung. Wählen Sie das Unterforum "Übungsblatt 9", wenn es konkret um eine Aufgabe geht oder das Unterforum "Kapitel 9"wenn es um eine Erklärung auf einer Vorlesungsfolie geht. Formulieren Sie Ihre Frage möglichst genau (z.B. "Was passiert, wenn man auf einen Zeiger p++ anwendet, der bislang auf den Anfang eines Feldes w gezeigt hat"). Am besten schreiben Sie ein konkretes Beispiel dazu (z.B. int w[5] = 1, 2, 3, 4, 5; int *p = w; p++;) und geben dem Thema anschließend einen sprechenden Titel (z.B. Umbiegen eines Zeigers auf ein Feld"). Einer der Tutoren oder Mitarbeiter wird zeitnah eine Antwort zu Ihrer Frage schreiben.

Aufgabe 33 *(Zeiger, 25 Minuten)*

a) (*, Benutzung von Zeigern, 15 Minuten)

1. Deklarieren Sie einen Zeiger `r`, der auf die dritte Komponente eines `int`-Feldes `w` zeigt.
2. Deklarieren Sie einen `int`-Zeiger `p`, der nirgendwohin zeigt.
3. Sei `n` eine Variable vom Typ `int`. Deklarieren Sie einen Zeiger `p` mit Bezugsvariable `n`.
4. Betrachten Sie die folgende Anweisung:
   ```
   int v[] = {5, 7, 9};
   ```
   Welchen Wert hat der Ausdruck `*(v + 2)`?
5. Betrachten Sie die folgenden Anweisungen:
   ```
   int v[] = {5, 7, 9};
   int *p = v;
   ```
   Welchen Wert hat der Ausdruck `*(p++)`?
6. Betrachten Sie die folgenden Anweisungen:
   ```
   int v[] = {5, 7, 9};
   int *p = v;
   ```
   Welchen Wert hat der Ausdruck `++(*p)`?

7. Betrachten Sie die folgenden Anweisungen:
```
int v[] = {5, 7, 9};
int *p = v;
```
Welchen Wert hat der Ausdruck `*(++p)`?

8. Betrachten Sie die folgenden Anweisungen:
```
char v[] = "Hallo";
v[2] = '\0';
printf("%s", v);
```
Was ist die Ausgabe der `printf`-Anweisung?

9. Betrachten Sie die folgenden Anweisungen:
```
char v[] = "Hallo";
printf("%s", v + 2);
```
Was ist die Ausgabe der `printf`-Anweisung?

10. In die folgenden Anweisungen hat sich ein Fehler in der Benutzung von Zeigern oder Feldern eingeschlichen:
```
char v[] = "Informatik";
int *p = v;
```
Welcher?

11. In die folgenden Anweisungen hat sich ein Fehler in der Benutzung von Zeigern oder Feldern eingeschlichen:
```
int v[20];
int *p = v;
++(*p);
```
Welcher?

12. In die folgenden Anweisungen hat sich ein Fehler in der Benutzung von Zeigern oder Feldern eingeschlichen:
```
char v[] = "Informatik";
char w[] = v;
```
Welcher?

13. In die folgenden Anweisungen hat sich ein Fehler in der Benutzung von Zeigern oder Feldern eingeschlichen:
```
char *p;
*p = '5';
```
Welcher?

14. In die folgenden Anweisungen hat sich ein Fehler in der Benutzung von Zeigern oder Feldern eingeschlichen:
```
char *p = NULL;
*p = '5';
```
Welcher?

15. In die folgenden Anweisungen haben sich zwei Fehler in der Benutzung von Zeigern oder Feldern eingeschlichen:
```
char v;
char *p = v;
p[1] = '5';
```
Welche?

b) (*, Variablen im Arbeitsspeicher, 10 Minuten)
Fertigen Sie jeweils zu den vorgegebenen Anweisungen eine Arbeitsspeicher-Skizze nach folgenden Vorgaben an. Die Skizze soll die von den benutzten Variablen belegten Speicherbereiche und deren Dateninhalte zeigen.

- Zeichnen Sie eine Speicherzelle pro Datenwert für die Datentypen `char`, `int`, `double` und adresswertige Datentypen, unabhängig vom Speicherbedarf dieser Datentypen.
- In die Speicherzellen schreiben Sie die Datenwerte, falls diese vom Typ `char`, `int` oder `double` sind.
- Speicherzellen mit undefinierten Werten lassen Sie leer.
- Werte von Zeigern geben Sie wie im Skript als Pfeile an. Verwenden Sie keine konkreten Adresswerte.
- Zwischen nicht zwingend zusammenhängenden Speicherbereichen lassen Sie einen Abstand.

1. Betrachten Sie die folgenden Anweisungen:
```
double v[2];
double *p = v;
*p = 3.1;
++p;
```
Skizzieren Sie die Belegung des Arbeitsspeichers nach Abarbeitung dieser Anweisungen.

2. Betrachten Sie die folgenden Anweisungen:
```
char w[10];
char *p = strcpy(w, "Hallo");
```
Skizzieren Sie die Belegung des Arbeitsspeichers nach Abarbeitung dieser Anweisungen.

3. Betrachten Sie die folgenden Anweisungen:
```
char w[10];
strcpy(w, "Hallo");
char *p = w + strlen(w);
```
Skizzieren Sie die Belegung des Arbeitsspeichers nach Abarbeitung dieser Anweisungen.

4. Betrachten Sie die folgenden Anweisungen:
```
char w[10];
char *p = strcpy(w, "Hallo");
while (*p != 'H') {++p;}
```
Skizzieren Sie die Belegung des Arbeitsspeichers nach Abarbeitung dieser Anweisungen.

5. Betrachten Sie die folgenden Anweisungen:
```
char w[10];
char *p = strcpy(w, "Hallo");
while (*p != 'l' && *p != '\0') {++p;}
```
Skizzieren Sie die Belegung des Arbeitsspeichers nach Abarbeitung dieser Anweisungen.

Aufgabe 34 *(Call by Reference, 20 Minuten)*

Erstellen Sie in jeder Teilaufgabe ein kompilierbares und ausführbares C-Programm mit einer `main`-Funktion und weiteren Funktionen nach folgenden Vorgaben. Kompilieren Sie Ihr Programm mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` und führen Sie es aus.

a) (**, 6 Minuten)

- Implementieren Sie ohne Benutzung von Bibliotheksfunktionen eine Funktion

  ```
  void increment_if_zero(int *x, int *y)
  ```

  die den an der übergebenen Adresse x gespeicherten Wert um 1 erhöht, wenn der an der übergebenen Adresse y gespeicherte Wert 0 ist.

- Legen Sie in der `main`-Funktion drei `int`-Variablen mit den Werten `-1`, `5` und `0` an und versuchen Sie, den Wert der Variable mit dem Wert -1 mit der erstellten Funktion und den anderen beiden Variablen zu inkrementieren. Geben Sie dabei den Wert der (nicht) inkrementierten Variable vor und nach dem Funktionsaufruf aus. Sie müssen **keine** Fehlerbehandlung für `NULL`-Fehler implementieren.

b) (**, 8 Minuten)

- Implementieren Sie ohne Benutzung von Bibliotheksfunktionen eine Funktion

```
int multiples_of_x(int n, int x, int *lower, int *greater)
```

  die für zwei nicht-negative Zahlen `n` und `x` die die größte durch `x` teilbare ganze Zahl berechnet, die kleiner oder gleich `n` ist, und an der Adresse `lower` speichert. Außerdem berechnet die Funktion die kleinste durch `x` teilbare ganze Zahl, die größer als `n` ist und speichert sie an der Adresse `greater`. Wird für `lower` oder `greater` der Wert `NULL` übergeben, so findet keine Berechnung statt und die Funktion gibt `0` zurück. Konnten Ergebnisse berechnet werden, gibt die Funktion den Wert `1` zurück.

- *Beispiel: Wird die Funktion mit dem Wert* 20 *für* `n` *und* 7 *für* `x` *aufgerufen, so wird an der Adresse* `lower` *der Wert* 14 *gespeichert und an der Adresse* `greater` *der Wert* 21.

- Testen Sie Ihre Implementierung in der `main`-Funktion mit verschiedenen Werten und geben Sie die Ergebnisse aus.

- Zu Übungszwecken ist es hier nicht erlaubt, Zeiger-Variablen in der `main`-Funktion zu deklarieren.

c) (**, 8 Minuten)

- Implementieren Sie ohne Berücksichtigung von Pufferfehlern (`EOF`) eine Einlesefunktion

```
int read_percentage(int *percentage)
```

  die vom Benutzer eine ganze Zahl zwischen einschließlich `0` und `100` einliest und an der übergebenen Adresse `percentage` speichert. Bei gültigen Eingaben soll die Funktion `1` zurückgeben. Bei ungültigen Eingaben soll die Funktion bei Bedarf den Eingabepuffer leeren und den Wert `0` zurückgeben.

- Testen Sie in der `main`-Funktion die obige Einlesefunktion, indem Sie eine Benutzereingabe einlesen und auf der Kommandozeile ausgeben, ob die Eingabe gültig war oder nicht und im positiven Fall zudem die eingelesene Zahl ausgeben.
  Zu Übungszwecken sei es hier nicht erlaubt eine Zeiger-Variable in der `main`-Funktion zu deklarieren.

Aufgabe 35 *(Zeichenketten, 24 Minuten)*

Erstellen Sie in jeder Teilaufgabe ein kompilierbares und ausführbares C-Programm mit einer `main`-Funktion und weiteren Funktionen nach folgenden Vorgaben. Kompilieren Sie Ihr Programm mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` und führen Sie es aus.

a) (**, 6 Minuten)

- Implementieren Sie ohne Benutzung von Bibliotheksfunktionen eine Funktion

```
char *str_rchr(const char *cs, int c)
```

  die sich wie die Bibliotheksfunktion `strrchr` verhält.

- Testen Sie Ihre `str_rchr`-Funktion in der `main`-Funktion mit folgenden Aufrufen:
  ```
  str_rchr("", 'g')
  str_rchr("", '\0')
  str_rchr("Schokolade", 'o')
  str_rchr("Schokolade", 'x')
  ```

b) (**, 8 Minuten)

- Implementieren Sie ohne Benutzung von Bibliotheksfunktionen eine Funktion

  ```
  char *str_str(const char *cs, const char *ct)
  ```

  die sich wie die Bibliotheksfunktion `strstr` verhält.
- Testen Sie Ihre `str_str`-Funktion in der `main`-Funktion mit folgenden Aufrufen:
  ```
  str_str("Schokolade", "")
  str_str("", "")
  str_str("Schokolade", "Info")
  str_str("Schokolade", "lade")
  ```

c) (**, 8 Minuten)
Benutzen Sie geeignet die Bibliotheksfunktion `strtok`, um nach folgenden Vorgaben Uhrzeiten in ihre Bestandteile für Stunden, Minuten und Sekunden zu zerlegen:

- In den Uhrzeiten sollen folgende Trennzeichen erlaubt sein: `':'`, `'-'` und `'/'`.
- In einer Uhrzeitsangabe dürfen unterschiedliche Trennzeichen vorkommen, oder auch mehrere Trennzeichen direkt hintereinander.
- Die Uhrzeitangaben enthalten zuerst die Stunden, dann die Minuten und dann die Sekunden.
- Besteht die Uhrzeitangabe nicht aus drei Teilen, so geben Sie `"Formatfehler!"` aus.
- Besteht die Uhrzeitangabe aus drei Teilen, so speichern Sie Stunden, Minuten und Sekunden jeweils in einer separaten Zeichenkette und geben diese aus. Das Format von Stunden, Minuten und Sekunden soll nicht überprüft werden. Es muss also beispielsweise nicht überprüft werden, dass es sich bei Minuten um eine Zahl zwischen `0` und `59` handelt.

Übergeben Sie eine Uhrzeitangabe per Kommandozeilenparameter an Ihr Programm.

Testen Sie Ihr Programm für folgende Uhrzeitangabe:
```
"12:30:14"
"15-07-39"
"8:30/12"
"7-01-34-23"
"01/12"
"13//12//11"
```

Aufgabe 36 *(Dynamische Speicherverwaltung, 20 Minuten)*

Erstellen Sie ein kompilierbares und ausführbares C-Programm mit einer `main`-Funktion und weiteren Funktionen nach folgenden Vorgaben. Kompilieren Sie Ihr Programm mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` und führen Sie es aus.

a) (**, 5 Minuten)
Implementieren Sie eine Funktion

```
int *array_d_copy(int v[], int n)
```

die die ersten `n` Komponenten von `v` kopiert und den Speicherplatz für die Kopie dabei dynamisch reserviert. Sie können davon ausgehen, dass `v` mindestens `n` Einträge enthält. Die Funktion soll im Erfolgsfall einen Zeiger auf die Kopie und sonst `NULL` liefern.

b) (**, 6 Minuten)
Implementieren Sie eine Funktion

```
int *array_d_shuffle(int v[], int w[], int n)
```

die `v` abwechselnd mit den ersten `n` Komponenten von `v` und `w` befüllt und den dafür benötigten Speicherplatz dabei dynamisch anpasst. Sie können davon ausgehen, dass `v` und `w` jeweils mindestens `n` Einträge enthalten. Die Funktion soll im Erfolgsfall eine Zeiger auf das geänderte `v` und sonst `NULL` liefern.
Beispiel:
Der Aufruf `array_d_shuffle(v, w, 3)` soll für die Felder `v = {1, 2, 3}` und `w = {4, 5, 6}` einen Zeiger auf ein Feld `{1, 4, 2, 5, 3, 6}` liefern.

c) (**, 9 Minuten)
Testen Sie die obigen Funktionen in der `main`-Funktion nach folgenden Vorgaben:

- Legen Sie ein Feld an und initialisieren es. Kopieren Sie dieses Feld mit `array_d_copy` und geben Sie die Kopie aus.
- Mischen Sie die ersten 4 Komponenten Ihres Feldes in Ihre Kopie hinein und geben Sie die Kombination der beiden aus. Achten Sie darauf, dass Felder nicht umgebogen werden können!
- Vergessen Sie keine notwendigen Fehlerbehandlungen und hinterlassen Sie keine Speicherlecks.