Programmieren in C++

SS 2022

Vorlesung 3, Dienstag 10. Mai 2022

(Grundlegende Konstrukte, Ncurses, mehr zu Make)

Prof. Dr. Hannah Bast
Professur für Algorithmen und Datenstrukturen
Institut für Informatik, Universität Freiburg

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Erfahrungen mit dem Ü2 — .h Dateien und Makefile
- Inhalt
  - Grundlegende Konstrukte — while, for, if, else, switch, …
  - Hinweise zum Ü3 — Beispielprogramm, ncurses
  - Globale Variablen — Deklaration mit "extern"
  - Mehr zu Make — Patterns, Variablen, Funktionen

Ü3: Eine einfach Implementierung des Spiels **Snake** (mit Konsolengrafik, die wir auch für das ÜB4 und ÜB5 brauchen)

Programmiertechnisch etwas anspruchsvoller als Ü1 und Ü2

Erfahrungen mit dem Ü2 1/3

UNI FREIBURG

- Zusammenfassung / Auszüge
  - Für die meisten sehr gut machbar + relativ einfaches ÜB, da man viel aus der Vorlesung nachmachen konnte
  - Vielen fanden den Stoff der Vorlesung sehr interessant
  - Für einige geht die Vorlesung etwas zu schnell, aber es gibt ja die Aufzeichnung, die meisten finden das Tempo aber gut
  - Forum wird intensiv genutzt + schnelle Antworten → **super!**

    "Meine Begeisterung für die Kommandozeile, Vim und C++ wächst aktuell exponentiell ☺"

    "Die dunkle Macht heißt nicht Sauron sondern Aufschieberitis. Bitte Tipps, wie ich diese angelernte Faulheit bekämpfe!"

    "Abschnitte in YouTube Videos: vielen Dank für die Mühe!"

Erfahrungen mit dem Ü2 2/3

UNI FREIBURG

- Wie geht es Ihnen nach zwei Jahren Pandemie

  Sentimentanalyse: ca. 50% positiv, 40% neutral, 10% negativ

  "Mir geht es gut nach zwei Jahren Pandemie"

  "Mir geht es schlecht nach zwei Jahren Pandemie"

  "Die 'interessanten' Bahn-Erlebnisse habe ich nicht so vermisst"

  "Präsenzvorlesungen mit > 80 Teilnehmenden jetzt sehr ungewohnt"

  "Ich bin müde, nur noch müde ..."

  "Ich lerne am besten autodidaktisch und jetzt ist Mathe auch online"

  "Nach zwei Jahren Pandemie fühle ich mich sozial inkompetent"

  "Gut, aber ich bin froh, dass das Ende in Sicht ist"

  "Haben mich in Therapie gebracht + sich verbunden fühlen erschwert"

  "Habe durch eine große WG ein schönes soziales Umfeld"

  "Gut, ich habe in der Zeit viel über mich gelernt"

Erfahrungen mit dem Ü2 3/3

UNI FREIBURG

- Wie geht es Ihnen nach zwei Jahren Pandemie

  "Bin überrascht, wie wenige Menschen noch Masken tragen drinnen"

  "Ein ständig bedrückender und sorgenbereitender Faktor im Alltag"

  "Habe mich an Home-Studium gewöhnt, am Anfang Kontakte gefehlt"

  "Deutlich schwerer sich zu motivieren, wenn man jeden Tag Zuhause ist"

  "Inzwischen komplett verwirrt, wenn Leute ohne Maske rumlaufen"

  "Nostalgie setzt ein, manche Online-Aktivitäten waren doch sehr lustig"

  "Welche Pandemie? Habe seit 12 Jahren meinen Keller nicht verlassen"

  "Nach zwei Jahren Pandemie plus Ukrainekrieg Lust auf einen anderen Kontinent auszuwandern. Gerade fast schlimmer als während der letzten zwei Jahre. Das Studium hat hier was Therapeutisches."

  "Online-Lehre enttäuschend, einige Profs so leicht wie möglich gemacht"

  "Gut … aber ich bin ja auch Informatiker, also wäre zu viel sozialer Kontakt auch nicht gesund"

Grundlegende Konstrukte 1/10

UNI FREIBURG

- Die elementaren Datentypen

int	ganze Zahl, typisch 4 Bytes -2^{31} .. $2^{31}-1$
long int	ganze Zahl, typisch 8 Bytes -2^{63} .. $2^{63}-1$
char	1 Byte bzw. ASCII Zeichen -128 .. 127
float	Fließkommazahl, 4 Bytes
double	Fließkommazahl, 8 Bytes
bool	zwei Werte: true (wahr) oder false (falsch)

Varianten: unsigned int, unsigned long int, unsigned char fangen bei 0 an, zum Beispiel unsigned char: 0 .. 255

Grund für float statt double: schneller + weniger Platzverbrauch … relevant falls sehr viele solcher Zahlen gespeichert werden oder bei sehr vielen Operationen

Grundlegende Konstrukte 2/10

UNI FREIBURG

- Variablen
  - **Benennung** in camelCase mit erstem Buchstaben klein

    In der Regel Wörter in Variablennamen **nicht** abkürzen

    Ausnahme: Variable wird in einem lokalen Kontext häufig benutzt (z.B. Laufvariable einer Schleife), dann ist auch ein kurzer Name ok oder sogar besser (z.B. i oder j oder c)
  - **Deklaration** vor der Benutzung ist Pflicht
  - **Initialisierung** bei der Deklaration ist optional, sonst beliebiger unbekannter Wert:

    ```
    int x;  // Has an unknown value after this.

    int y = 10; // Value 10 after this.
    ```

Grundlegende Konstrukte 3/10

UNI FREIBURG

- Ausdrücke
  - Im Wesentlichen beliebige geklammerte Ausdrücke mit den Operatoren + – * / % (modulo), z.B.

    17 * (x – y / 2) + 32 * x * y / (5 – numValues)
  - Vergleichsoperatoren : < > <= >= == !=
  - Für bool außerdem : && (und) || (oder) ! (nicht)
  - Dann gibt es noch die bitweisen Operatoren | und & und die Bitschiebeoperatoren << und >>

    Die brauchen wir jetzt noch nicht, sind aber manchmal nützlich und werden in einer späteren Vorlesung erklärt

Grundlegende Konstrukte 4/10

UNI FREIBURG

- Zuweisungen
  - Normale Zuweisung

```
i = j + 2;  // Left side must be a variable.
```

  - Abkürzungen für häufige Muster von Zuweisungen

```
++i;        // Identical to i = i + 1.
−−i;        // Identical to i = i − 1.

x +=3;      // Identical to x = x + 3.
x −=3;      // Identical to x = x − 3.
x *= 3;     // Identical to x = x * 3.
x /= 3;     // Identical to x = x / 3.
x %= 3;     // Identical to x = x % 3.
```

Grundlegende Konstrukte 5/10

UNI FREIBURG

- Konditionale Ausführung von Code

```
if (condition) {
  // Code block 1.
  ...
} else {
  // Code block 2.
  ...
}
```

  - Falls condition wahr ist, wird Code block 1 ausgeführt, sonst wird Code block 2 ausgeführt
  - Der else Teil kann auch fehlen, dann wird bei falscher condition an dieser Stelle gar kein Code ausgeführt

Grundlegende Konstrukte 6/10

UNI FREIBURG

- Konditionale Ausführung mit **switch**
  - Bei vielen einfachen Gleichheitsbedingungen, z.B.

```
int keycode = getch();
switch (keycode) {
   case KEY_UP:        --row; break;
   case KEY_DOWN:      ++row; break;
   case KEY_LEFT:      --col; break;
   case KEY_RIGHT:     ++col; break;
   default: …;  // If none of the cases match.
}
```

  - Den **default** Teil kann man auch einfach weglassen

Achtung: ohne das break wird auch der anschließende Code ausgeführt, das ist in der Regel nicht das, was man möchte

Grundlegende Konstrukte 7/10

UNI FREIBURG

- Der 3-Wege-Operator
  - Sehr nützlich, um bei einfachen Konditionalen ein if - else über mehrere Zeilen zu vermeiden:

    min = x < y **?** x **:** y; // The minimum of x and y.

  - Die allgemeine Form ist

    condition **?** expression1 **:** expression2

  - Der Wert des Ausdrucks ist expression1 wenn condition wahr ist und sonst expression2

    expression1 und expression2 müssen vom selben Typ sein

Grundlegende Konstrukte 8/10

UNI FREIBURG

- Schleifen: while und for

```
// Print the numbers from 1 to 10.
int i = 1;
while (i <= 10) {
  printf("%d\n", i);
  ++i;
}
```

  - Äquivalent dazu, aber kürzer und besser lesbar:

```
// Print the numbers from 1 to 10.
for (int i = 1; i <= 10; ++i) {
  printf("%d\n", i);
}
```

Grundlegende Konstrukte 9/10

UNI FREIBURG

- Konvention: for nur bei **einer** Schleifenvariablen
  - ... und relativ **einfacher** Abbruchbedingung, sonst while

```
// Valid but opaque, better use while!
for (int i = 0, int j = 10; i < j; ++i, −−j) {
  printf("%d %d\n", i, j);
}

// Equivalent while loop, longer but easier to understand.
int i = 0;
int j = 10;
while (i < j) {
  printf("%d %d\n", i, j);
  ++i;
  −−j;
}
```

Grundlegende Konstrukte 10/10

UNI FREIBURG

- Schleifen: break und continue
  - Schleife vorzeitig abbrechen: **break**
  - Eine Iteration überspringen: **continue**

```
// Read key, print if letter, stop when "Escape" key pressed.
while (true) {
   int keycode = getch();
   if (keycode == 27) { break; }
   if (keycode < 'a' || keycode > 'z') { continue; }
   printf("Keycode: %c\n", keycode);
}
```

  - Bei geschachtelten Schleifen: Abbruch aus der Schleife, in der das break steht, nicht auch aus den umschließenden Schleifen

Hinweise zum Ü3 1/5

UNI FREIBURG

- Snake
  - Aufgabe des Ü3 ist es, eine einfache Version von **Snake** zu implementieren:

    Eine "Schlange" fixer Länge bewegt sich mit 10 Pixel / Sek. über den Bildschirm, dabei steuerbar mit den Pfeiltasten

    Das Spiel ist zu Ende, wenn die Schlange auf den Bildschirmrand oder einen Teil von sich selber stößt

    Ziel ist es, diesen Moment möglichst lange herauszuzögern und bis dahin erleuchtet zu werden

  - Zur Unterstützung für das Übungsblatt implementieren wir heute zusammen ein paar einfache Animationen

    Konsolengrafik brauchen wir auch für das Ü4 und Ü5

Hinweise zum Ü3 2/5

UNI FREIBURG

- Ncurses … Initialisierung
  - Eine Bibliothek für erweiterte Ausgabe über die Konsole und Eingabe über die Tastatur / Maus

```
#include <ncurses.h>

initscr();                  // Initializiation.
cbreak();                   // Don't wait for RETURN.
noecho();                   // Don't echo key presses on screen.
curs_set(false);            // Don't show the cursor.
nodelay(stdscr, true);      // Don't wait until key pressed.
keypad(stdscr, true);       // For KEY_LEFT, KEY_UP, etc.
```

  - Beim Linken brauch man dann noch: -lncurses
  - Falls nicht installiert: sudo apt install libncurses-dev
  - Dokumentation: sudo apt install ncurses-doc dann geht z.B. man initscr

Hinweise zum Ü3 3/5

UNI FREIBURG

- Ncurses ... Funktionen
  - Schreiben an eine bestimmte Position

    mvprintw(y, x, "..."); es geht auch sowas wie mvprintw(y, x, "%d", 4)

    Achtung: die Koordinaten sind aus "Terminalsicht" = erst die Zeile (y-Koordinate), dann die Spalte (x-Koordinate)
  - Invers malen (Vorder- und Hintergrundfarbe vertauschen)

```
attron(A_REVERSE);    // Switch on reverse mode
attroff(A_REVERSE);   // Switch off reverse mode.
```

  - Verwendung von Farben

```
start_color();
init_pair(42, COLOR_BLUE, COLOR_WHITE);
ATTRON(COLOR_PAIR(42)); // Switch off with ATTROFF(COLOR_PAIR(42)).
```

Hinweise zum Ü3 4/5

UNI FREIBURG

- Ncurses … weitere Funktionen und Tipps
  - Damit die Änderungen auch erscheinen: refresh()
  - Aufräumen ganz am Ende (wichtig): endwin();
  - Zum Nichtstun für eine bestimmte Zeit

```
#include <unistd.h>

usleep(100'000);  // Do nothing for 100.000µs = 100ms.
```

  - Code der letzten gedrückten Taste

```
int key = getch();
if (key == KEY_UP) { … }  // Arrow up key pressed.
```

  - Globale Variablen für die Bildschirmgröße:

    LINES (Anzahl Zeilen) und COLS (Anzahl Spalten)

Hinweise zum Ü3 5/5

UNI FREIBURG

- Ncurses und Unit Tests
  - Achten Sie darauf, dass der Code für das **Bewegen** und für das **Malen** in verschiedenen Funktionen steht

    Für das Ü3 sind die Funktionen bereits so vorgegeben, dass das der Fall ist
  - Für die Funktionen, die nur malen, brauchen Sie keinen Unit Test zu schreiben (es wäre auch kompliziert)
  - Der Unit Test für die Initialisierung braucht auch nicht zu überprüfen, ob ncurses richtig initialisiert wurde (dito)
  - Für den Code in der **main** Funktion auch kein Unit Test

    Deswegen sollte in der **main** Funktion auch grundsätzlich so wenig Code wie möglich stehen

Globale Variablen 1/3

UNI FREIBURG

- Was + warum
  - Variablen, die außerhalb einer Funktion definiert sind, nennt man **globale Variablen**

```
int x;

void someFunction() {
  // x can be used here.
  ...
}
```

  - Globale Variablen kann man im Prinzip überall im Code benutzen, auch in anderen Dateien

    Es ist dasselbe Prinzip wie bei unseren Funktionen bisher, siehe nächste Folien

Globale Variablen 2/3

UNI FREIBURG

- Wiederholung: Linken von Funktionen
  - Jede Funktion muss vor der Benutzung **deklariert** werden

    Üblicherweise in einer .h Datei, die dann in jeder .cpp Datei, in der die Funktion benötigt wird, inkludiert wird
  - Jede Funktion muss in genau einer Datei **definiert** (das heißt: implementiert) sein

    Die dazugehörige .o Datei oder Bibliothek muss dann beim Linken dabei sein

Globale Variablen 3/3

UNI FREIBURG

- Das gilt genauso für globalen Variablen
  - Die **Deklaration** geht dort mit dem Schlüsselwort extern

    ```
    extern int x;
    extern int y;
    ```

  - Die **Definition** dann wie gehabt ohne das extern:

    ```
    int x;
    int y;
    ```

  - Wenn eine globale Variable mit extern deklariert wurde und dann beim Linken nicht gefunden wird, kommt auch einfach

    "undefined reference to ..."

    (und bei Mehrfachdefinition: "multiple definition of ...")

Noch mehr zu Make 1/5

UNI FREIBURG

- Pattern-Regeln

```
%.o: %.cpp
        <command1>
        <command2>
        ...
```

  - Wird angewendet für jedes target, das zu %.o passt, z.B.

    make Snake.o

  - Das % auf der rechten Seite wird dann entsprechend ersetzt

    Im Beispiel durch "Snake"

Noch mehr zu Make 2/5

UNI FREIBURG

- Automatische Variablen (beim Match einer Pattern-Regel)

  $@ ist das konkrete target

  $* ist dieses target ohne Suffix

  $< ist die erste "dependency" nach dem target

  $^ sind alle "dependencies" nach dem target

  Beispiel:

```
%.o: %.cpp
        g++ -o $@ -c $^
```

  Dann wird bei make Snake.o ausgeführt:

```
g++ -o Snake.o -c Snake.cpp
```

Noch mehr zu Make 3/5

UNI FREIBURG

- Variablen

  CXX = g++

  Wie in einem C++ Programm, nur ohne Variablentyp, kann man dann an andere Stelle im Makefile so verwenden:

```
%.o: %.cpp
        $(CXX) -o $@ $^
```

Noch mehr zu Make 4/5

UNI FREIBURG

- Funktionen
  - Liste aller Dateien im aktuellen Ordner (durch Leerzeichen getrennt), die zu einem bestimmen Muster passen

    $(wildcard *.cpp)
  - Entfernen aller Suffixe einer Liste von Strings (Dateinamen)

    $(basename Snake.o SnakeMain.cpp SnakeTest.o)
  - Weitere Funktionen (Erklärung siehe Referenzen am Ende)

    $(filter %Main.cpp, <list of strings>)

    $(filter-out %Main.cpp, <list of strings>)

    $(addsuffix .o, <list of strings>)

Noch mehr zu Make 5/5

UNI FREIBURG

- Die main Funktion in der ...Test.cpp Datei
  - Brauchen wir, damit es überhaupt linkt und damit das Programm dann wie gewünscht alle Tests ausführt
  - Diese main Funktion ist aber immer **genau** dieselbe:

```
int main() {
  ::testing::InitGoogleTest();
  return RUN_ALL_TESTS();
}
```

  - Deswegen gibt es eine eigene Bibliothek, die nichts anderes enthält wie diese (bzw. eine sehr ähnliche) main Funktion
  - Die kann man einfach mit –lgtest_main dazulinken

Dann **muss** man die main Funktion in ...Test.cpp weglassen !

Literatur / Links

UNI FREIBURG

- Grundlegende Konstrukte in C++
  - http://www.cplusplus.com/doc/tutorial/variables/
  - http://www.cplusplus.com/doc/tutorial/operators/
  - http://www.cplusplus.com/doc/tutorial/control
- Makefile Patterns, Variablen, Funktionen
  - http://www.gnu.org/software/make/manual/make.html

    Kapitel: "Pattern Rules", "Automatic Variables", "Functions"
- Ncurses, man pages
  - sudo apt-get install ncurses-doc
  - man ncurses oder man mvprintw oder man getch oder ...