ETH zürich

Übungsstunde W06

Informatik (RW) – HS 23

Übersicht

Heutiges Programm

Follow-up
Feedback zu **code** expert
Ziele
assert
PRE und POST
Functions
Exam Question
Stepwise Refinement
Outro

`rwko.ch/lily`

1. Follow-up

Follow-up

- Was ist *Binary Expansion*?
  - Das[1]
- Wenn ihr mir während der Übungsstunde fragen stellt, dann bitte schickt mir noch eine Follow-up-mail damit ich euren Namen weiss und direkt antworten kann

[1] https://lec.inf.ethz.ch/math/informatik_cse/2023/slides/lecture5.en.handout.pdf

2. Feedback zu **code** expert

Allgemeines zu **code** expert

Allgemeines zu **code** expert

- Bitte behaltet den Code innerhalb der grauen Linie
  - Nutzt dafür einfach mehrzeilige Kommentare
- Es gibt fast immer einen besseren Ansatz; fühlt euch nicht schlecht, wenn ihr ihn nicht auf Anhieb hattet
- `n = n + 1` und `n += 1` ist nicht sehr typisch für C++, schreibt lieber `n++`
- Löscht ruhig die `<Insert your answer here, within the comment block>` beim Beantworten der Fragen

Allgemeines zu **code** expert

- Fast alle Antworten waren viel ausführlicher als nötig
- Was $\neq$ Wie
  - Wenn ihr gefragt werdet, *was* ein Code-Snippet macht, dann fängt nicht an zu erklären *wie* er es macht
  - Tipp: Wenn ihr Variablennamen erwähnt, erklärt ihr wahrscheinlich wie etwas funktioniert und nicht was es macht
- Bitte lasst Abstände zwischen Operatoren und Variablen
  - also nicht so: `int b=a+2*x-(5/6)+(9+1*a)`
  - sondern so: `int b = a + 2 * x - (5 / 6) + (9 + 1 * a)`
- Wenn mehrere Aufgaben sehr ähnlich waren, gab es meist nur ausführliches Feedback zu einer davon
- Ich kann Verbesserungen/Vorschläge in euren Code reinschreiben und ihr könnt sie sehen unter *View Submission*

Fragen/Unklarheiten?

3. Ziele

Ziele

- ☐ assert-statements zum Debuggen benutzen können
- ☐ Gute PRE- und POST-Conditions schreiben können
- ☐ Aufgaben mittels *Stepwise Refinement* (dt. *Schrittweise Weiterentwicklung*) lösen können

4. assert

assert

- Wir werden exit codes sehen, mehr Infos hier [2]

[2] `https://lec.inf.ethz.ch/ifmp/2023/guides/debugging/exit_codes.html`

assert

Frage

- Frage: Wofür sind asserts nützlich?

assert

Frage

- Frage: Wofür sind asserts nützlich?

Mögliche Antworten

- Um herauszufinden, wo genau ein Fehler passiert
- In langen Programmen, um den Überblick zu bewahren
- Falsche User Inputs sofort erkennen -> hilft, undefined behavior zu vermeiden
- Als eine Art der Code Dokumentation

5. PRE und POST

PRE und POST Conditions

```
// PRE:  describes accepted input
// POST: describes expected output
int yourfunction(int a, int b){
    ...
}
```

PRE und POST Conditions

Frage

Was wären hier sinnvolle Conditions?

```
// PRE:
// POST:
double area(double height, double length){
    return height*length;
}
```

PRE und POST Conditions

Frage

Was wären hier sinnvolle Conditions?

```
// PRE:
// POST:
double area(double height, double length){
    return height*length;
}
```

Sie müssen nicht sehr detailiert sein, sie sollten beschreiben, was die Funktion erwartet und was sie ausgeben wird (wenn der Input erwartungsgemäss war)

Fragen/Unklarheiten?

PRE und POST Conditions I (Lösung)

Mögliche Lösung

```
// PRE:  (not needed)
// POST: return value is maximum of {i, j, k}
double f(double i, double j, double k){
    if(i > j){
        if(i > k){return i;}
        else {return k;}
    } else {
        if(j > k){return j;}
        else {return k;}
    }
}
```

PRE und POST Conditions II

Bestimme sinnvolle PRE- und POST-conditions für die folgende Funktion

```
// PRE:  ???
// POST: ???
double g(int i, int j){
    double r = 0.0;
    for(int k = i; k <= j; k++){
        r += 1.0 / k;
    }
    return r;
}
```

PRE und POST Conditions II (Lösung)

Mögliche Lösung

```
// PRE:  0 not in [i, j] and i <= j <= INT_MAX
// POST: return value is the sum 1/i + 1/(i+1) + ... + 1/j
double g(int i, int j){
    double r = 0.0;
    for(int k = i; k <= j; k++){
        r += 1.0 / k;
    }
    return r;
}
```

6. Functions

Output?

```
int f(int i){
    return i * i;
}

int g(int i){
    return i * f(i) * f(f(i));
}

int h(int i){
    std::cout << g(i) << "\n";
}
// ...
```

```
// ...
int main(){
    int i;
    std::cin >> i;
    h(i);
    return 0;
}
```

Was wird (mögliche Over- und Underflows vernachlässigt) der Output sein?

Output?

```
int f(int i){
    return i * i;
}

int g(int i){
    return i * f(i) * f(f(i));
}

int h(int i){
    std::cout << g(i) << "\n";
}
// ...
```

```
// ...
int main(){
    int i;
    std::cin >> i;
    h(i);
    return 0;
}
```

Was wird (mögliche Over- und Underflows vernachlässigt) der Output sein? **Lösung**: i^7

Fehlersuche

```
double f(double x){
    return g(2.0 * x);
}

double g(double x){
    return x % 2.0 == 0;
}

double h(double x){
    std::cout << result;
}
// ...
```

```
// ...
int main(){
    double result = f(3.0);
    h();

    return 0;
}
```

Finde 3 Fehler in diesem Programm.

Fehlersuche (Lösung)

1. `g()` ist `f()` nicht bekannt, da der Scope von `g()` erst später anfängt
2. Es gibt keinen `%`-Operator für `double`
3. `h()` “sieht” die Variabel `result` nicht, da sie nicht im gleichen Scope ist

Anzahl Divisoren

Schreibe eine Funktion `number_of_divisors()`, die `int n` als Argument entgegennimmt und die Anzahl Divisoren von n zurückgibt (inklusive 1 und n)

```
// PRE:  0 < n < MAX_INT
// POST: returns number of divisors of n (incl. 1 and n)
unsigned int number_of_divisors(int n){
    // ...
}
```

Beispiel
6 hat 4 Divisoren, nämlich $1, 2, 3, 6$

Anzahl Divisoren (Lösung)

```
// PRE:  0 < n < MAX_INT
// POST: returns number of divisors of n (incl. 1 and n)
unsigned int number_of_divisors(int n){
    assert(n > 0);
    unsigned int counter = 0;

    for (int i = 1; i <= n; ++i){
        if(n % i == 0){
            counter++;
        }
    }

    return counter;
}
```

Fragen/Unklarheiten?

7. Exam Question

Sehr prüfungsrelevant!

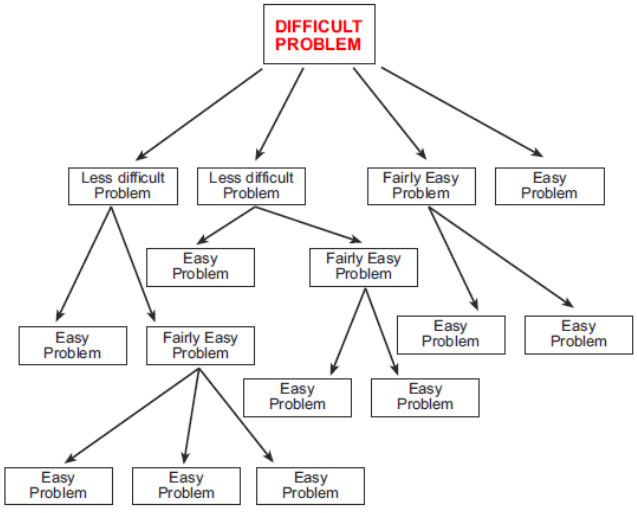
- Das ist eine echte Prüfungsaufgabe aus dem Jahr 2022
- Öffnet die Aufgabe “[Exam 2022.02 (MAVT + ITET)] Decimal to arbitrary base” auf **code** expert
- Besprecht die Aufgabe und euren Lösungsansatz mit euren Nachbar:innen

Sehr prüfungsrelevant!

- Das ist eine echte Prüfungsaufgabe aus dem Jahr 2022
- Öffnet die Aufgabe “[Exam 2022.02 (MAVT + ITET)] Decimal to arbitrary base” auf **code** expert
- Besprecht die Aufgabe und euren Lösungsansatz mit euren Nachbar:innen
- Löst die Aufgabe

8. Stepwise Refinement

Grundidee


DIFFICULT PROBLEM
Less difficult Problem
Less difficult Problem
Fairly Easy Problem
Easy Problem
Easy Problem
Fairly Easy Problem
Easy Problem
Fairly Easy Problem
Easy Problem
Easy Problem
Easy Problem
Easy Problem
Easy Problem
Easy Problem
Easy Problem

Stepwise Refinement

Code Example “Perfect Numbers” on **code** expert

Write a program that counts how many perfect numbers exist in the range $[a, b]$. Please use stepwise refinement to develop a solution to this task that is divided into meaningful functions. We provide a function `is_perfect` in `perfect.h` that checks if a given number is perfect.

A number $n \in \mathbb{N}$ is called perfect if and only if it is equal to the sum of its proper divisors. For example:

- $28 = 1 + 2 + 4 + 7 + 14$ is perfect
- $12 \neq 1 + 2 + 3 + 4 + 6$ is not perfect

Stepwise Refinement

- *nicht sofort programmieren!*
- identifiziert die einfacheren Teilprobleme

Stepwise Refinement

- *nicht sofort programmieren!*
- identifiziert die einfacheren Teilprobleme
- welche Teilprobleme konntet ihr identifizieren?

“Problembaum”

Wie viele vollkommene Zahlen (engl. perfect numbers) gibt es?

Lösung zu “Perfect Numbers”

```
// PRE:
// POST:
bool is_perfect(unsigned int number) {
  unsigned int sum = 0;
  for (unsigned int d = 1; d < number; ++d) {
    if (number % d == 0) {
      sum += d;
    }
  }
  return sum == number;
}
```

Lösung zu “Perfect Numbers”

```
#include <iostream>
#include "perfect.h"

// PRE:
// POST:
unsigned int count_perfect_numbers(unsigned int a, unsigned int b) {
  unsigned int count = 0;
  for (unsigned int i = a; i <= b; ++i) {
    if (is_perfect(i)) {
      count++;
    }
  }
  return count;
}

...
```

Lösung zu “Perfect Numbers”

```
...

int main () {
  // input
  unsigned int a;
  unsigned int b;
  std::cin >> a >> b;

  // computation and output
  unsigned int count = count_perfect_numbers(a, b);

  // output
  std::cout << count << std::endl;

  return 0;
}
```

Fragen/Unklarheiten?

9. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche!