Exercise Session

Week 10

Adel Gavranović
agavranovic@student.ethz.ch

Overview

▸ polybox for session material ▸ mail to TA

Today's Topics

Introduction

Self-Assessment

Classes and Structs

Overloading

Intro

- I'm in the process of getting a new internet connection, so if I suddenly disappear that might be the reason

Follow-up

- **”Red icon even if on submission was on time”:** The [code]expert team knows about it and is working on it. Don’t worry about your bonus, if you solved the exercise well you will get your grade increase.

Comments on last [code]expert Exercises

- Don’t bother cheating. They *will* catch you

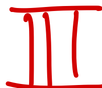
Questions or Comments re: Exercises?

Learning Objectives Checklist

Now I...

- ☐ can define classes
- ☐ can overload operators for the classes I've written

Self-Assessment



- log into Moodle
- on my command, start the self assessment
- should take around 20 minutes
- we'll have a quick break and then discuss the Self-Assessment after the break

Characters

Betrachten Sie die folgende Schleife und geben Sie ihre Ausgabe direkt in dem dafür vorgesehenen Kästchen an. Nehmen Sie an, dass Buchstaben gemäss dem ASCII-Code sortiert sind!

Consider the following loop and write its output directly in the provided box. Assume that letters are ordered according to the ASCII code!

```
for (char c = 'F'; c < 'R'; c += 3) {
  std::cout << c;<< " ";
}
```

char 1

FILO F,I,L,O

Ausgabe output: F I L O

```
Tree = '(' Branches ')'.
Branches = Branch { Branch }.
Branch = Length | Length Tree.
Length = unsigned int.
```

implement these Rules

```
#include<iostream>
int main() {
  int length = Tree(std::cin);
  std::cout << "Maximal path length= " << length << \n";
  return 0;
}

// POST: leading whitespace characters are extracted from is, and the
// first non-whitespace character is returned (0 if there is none)
char lookahead (std::istream& is);

// Tree = '(' Branches ')'.
// Returns length of longest outgoing path.
int Tree (std::istream& is){
  char c;
  is >> c; assert(c=='(');
  int length = Branches(is);
  is >> c; assert(c==')');
  return length;
}

// Branches = Branch { Branch }.
int Branches (std::istream& is){
  int length = Branch(is);
  while (lookahead(is) != ')'){
    int bLength = Branch(is);
    if (bLength > length) length = bLength;
  }
  return length;
}

// Branch = Length | Length Tree.
int Branch(std::istream& is){
  int length = Length(is);
  if (lookahead(is) == '(') return length + Tree(is);
  return length;
}

// Length = unsigned int.
// Returns integer present at input stream
int Length (std::istream& is);
```

if (...){ {...} }

Seien x und y zwei Folgen mit Längen n und m, $x = (x_1, x_2, ..., x_n)$ und $y = (y_1, y_2, ... , y_m)$. Wir definieren den Reissverschluss von x und y als

Let x and y be two sequences of lengths n and m, $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ... , y_m)$. We define the zip of x and y as

$$zip(x, y) = (x_1, y_1, x_2, y_2, ... , x_{min(n,m)}, y_{min(n,m)})$$

Gesucht ist eine Funktion, die zwei Folgen ganzer Zahlen x und y als Zeigerbereiche erhält und zip(x, y) ausgibt. Hier ist ein Beispiel:

We are looking for a function that re- ceives two integer sequences x and y as ranges and outputs zip(x, y). Here is an example:

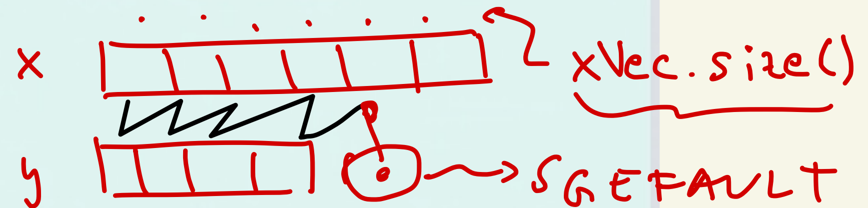
x = (1, 3, 5, 7, 9, 11), y = (2, 4, 6, 8) → zip(x, y) = (1, 2, 3, 4, 5, 6, 7, 8).

```
void zip (const std::vector<int>& xVec, const std::vector<int>& yVec)
{
   unsigned int xInd = 0 ;

   unsigned int yInd = 0 ;

   while ( (xInd < xVec.size()) && (yInd < yVec.size()) ) {

      std :: cout << xVec.at(xInd) << " " << yVec.at(yInd);
<< " ";
      ++xInd;
      ++yInd;
   }
}
```




```
int main ()
{
   std::vector<int> x = {1, 3, 5, 7, 9, 11};
   std::vector<int> y = {2, 4, 6, 8};
   zip ( x, y ); // 1 2 3 4 5 6 7 8
                    2 1 4 3    (zip y, x)
}
```

Post-Self-Assessment Discussion

- what did you find particularily hard?
- which tasks should we go over again?
- what was easy?
- how is exam-prepp going?

Questions?

`Struct` vs `Class`

Difference

The only difference between them is their default *visibility*.

`Struct`	public (”visible”)
`Class`	private (”invisible”)

You can change the visibility of your members in both classes and structs by specifying it with the keywords `private:` or `public:` respectively.

When should you use what?

Doesn’t really matter, as long as you get the visibility right.
Recommendation: use `class` for ”complicated stuff” and `struct` for ”bundles of data”.

Questions?

Function overloading

This part covers the question ”How does the computer know which function I want to call, if two functions have the same name?”
It *is* possible for two functions to have the same name, as long as the compiler has another way to differentiate between them with the help of the following criteria.

Viable Criteria

- number of function parameters
- type of function parameters

Not Viable Criteria

- name of the function parameter
- return-type of the function

Function overloading

multiply(a , b)

a * b

Why do we even care if two functions can have the same name?

Because we can then use ”operator overloading” much more smoothly. Operators (such as $*$, +, =) are basically fancy looking functions. The operator ”$*$” for example can then have multiple meanings depending on what is given as an input argument. So it can perform ”normal” multiplication when given two `int` variables but does something else when we give it a certain `class`.

Exercise "Tribool"

Tribool: a bool, but with three values (`false`, `unknown`, `true`).
Here are the logic tables for the operators on tribools.

NOT(A)

A	¬A
F	T
U	U
T	F

AND(A,B)

A ∧ B		B: F	B: U	B: T
A	F	F	F	F
A	U	F	U	U
A	T	F	U	T

OR(A,B)

A ∨ B		B: F	B: U	B: T
A	F	F	U	T
A	U	U	U	T
A	T	T	T	T

F = FALSE, U = UNKNOWN, T = TRUE

Exercise ”Tribool”

This implementation stores the truth value in a `class` called ”Tribool” as a `private unsigned int`.

- why is this a good idea?
- how could we store the information (truth value)?
- we’re going to solve this exercise together

Exercise "Tribool" Concepts

We're going to see all of these concepts in action when doing this exercise:

- classes ✓
- visibility ✓
- operator overloading
- declaration vs definition
- out-of-class definitions
- `const` functions
- constructors

private vs. public

&&

decl: foo(...); → .h

def: foo(...){ ... } → .cpp

Tribool:: ...

won't change values inside of class/struct

fancy functions that get called right when initializing

Exercise ”Tribool”

- **Step 1:** We’re going to implement the first constructor together and you will try to implement the second one on your own
- **Step 2:** try the second step by yourself too
- **Step 3:** we’re first going to discuss how to implememt this in a very clever way and then you are going to write it on your own

Questions?

M.F.: can't we make the std::string function parameters const-references as well?

A.G.: Yes! (Just declare and define them properly in the .h and .cpp files.