Informatik I

Uebung 8

Gruppe A, ETZ E91

Patrick Boenzli

dev.orxonox.net/wiki/boenzlip/inf1

Lernziele

Die Studenten:

- kennen die Grundlagen von C++
- wissen was eine **Rekursion** ist und wie man eine solche implementiert
- kennen den Algorithmus “**Sortieren durch Einfuegen**”

Rekursive Algorithmen

a) **Begriffserklaerung**:

- **Alternative** zu **iterativen** Verfahren
- bei der Rekursion **ruft sich** eine Funktion **selber** wieder **auf**
- **Abbruchsbedingung** beendet den rekursiven Abstieg

Rekursive Algorithmen

b) Beispiel: Fibonacci Algorithmus

Fibonacci-Reihe:

`fib(n) = fib(n-1) + fib(n-2)`

```
fib(2) = 1
fib(1) = 1
```

Wie kann die Fibonacci-Reihe implementiert werden?

Rekursive Algorithmen

c) Beispiel: Fibonacci Algorithmus - Iterativ

```
int fib( int i )  {
    int  f0 = 0; //previous fib number
    int  f1 = 1; //current fib number
    int  tmp;

    if ( i == 0 ) return 0;
    for( int j = 2; j <= i; j++ ) {
      tmp = f1;
      f1 += f0;
      f0 = tmp;
    }
    return f1;
  }
```

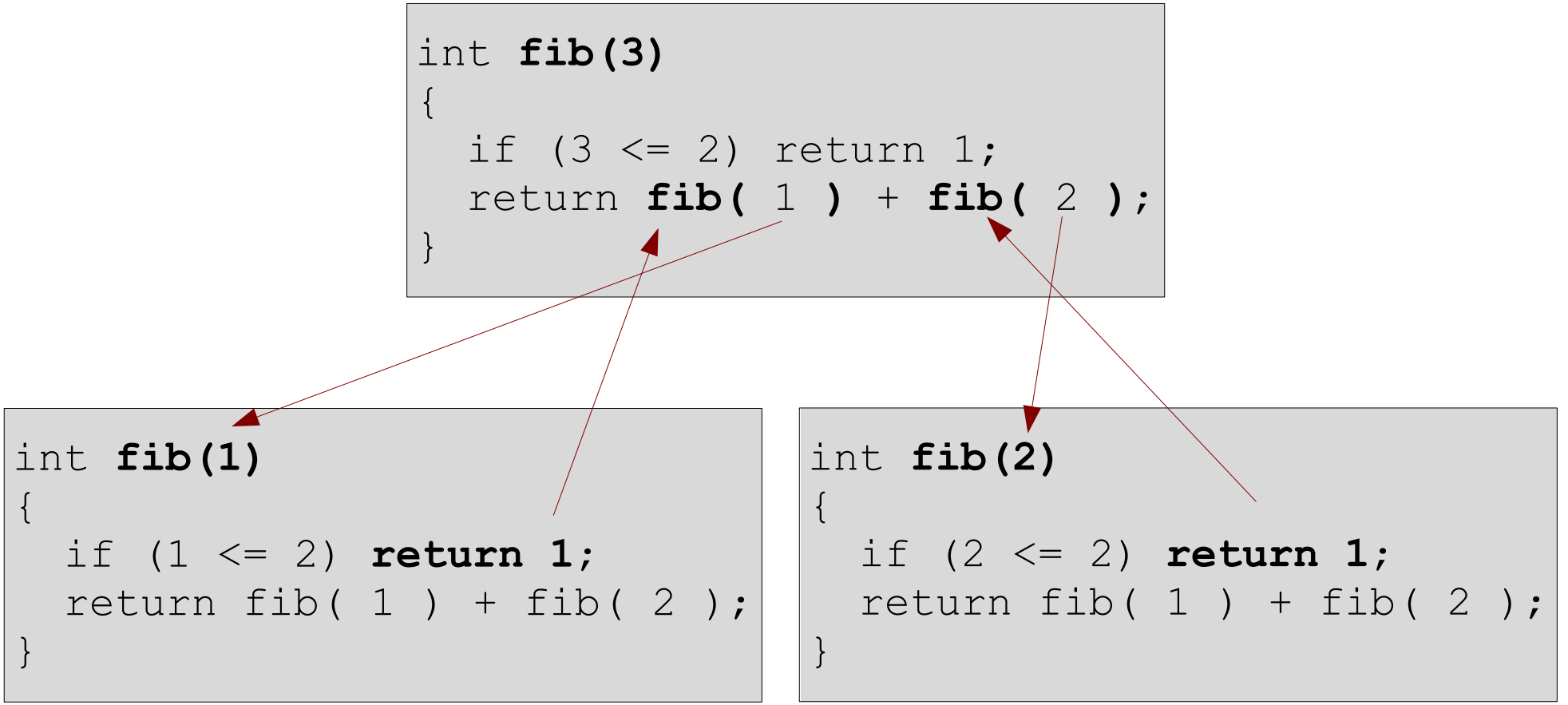
Rekursive Algorithmen

d) Beispiel: Fibonacci Algorithmus - Rekursiv

```
int fib( int n )
{
  if (n <= 2) return 1;
  return fib( n-2 ) + fib( n-1 );
}
```

Rekursive Algorithmen

e) Beispiel: Fibonacci Bildhaft, n = 3



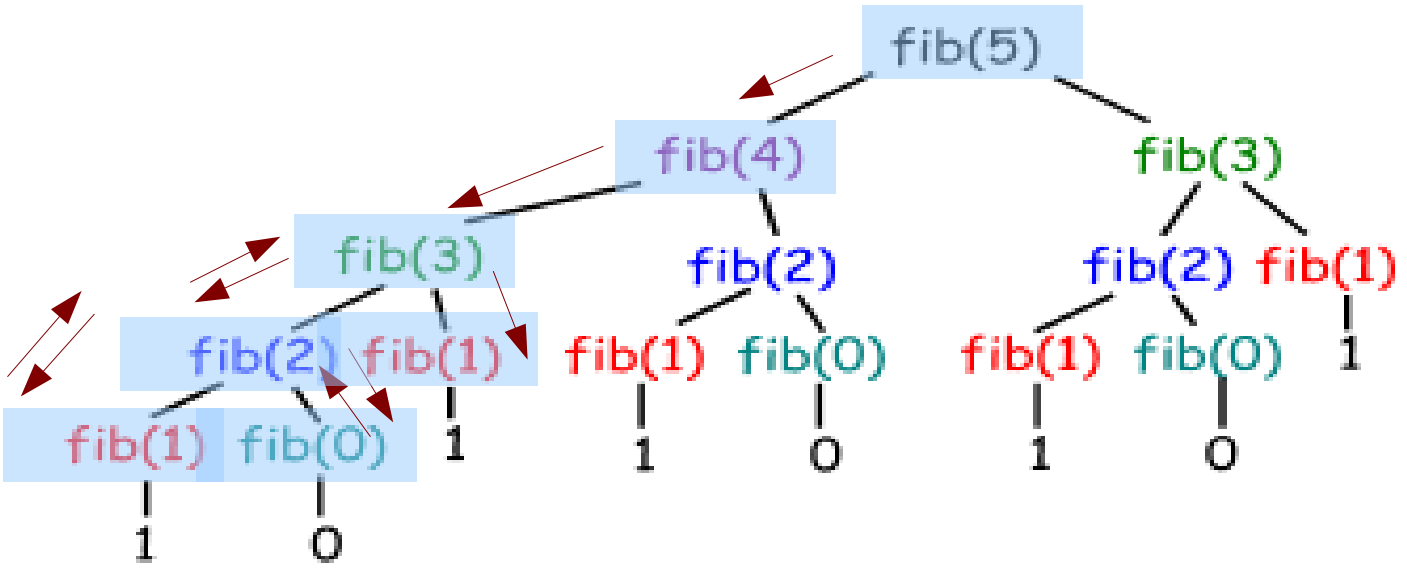
Rekursive Algorithmen

d) Beispiel: Fibonacci – Call Stack

```
int fib( int n ) {
  if (n = 1) return 1;
  else if (n <= 0) return 0;
return fib( n-2 ) + fib( n-1 );
}
```

Ausfuehren der Funktion:

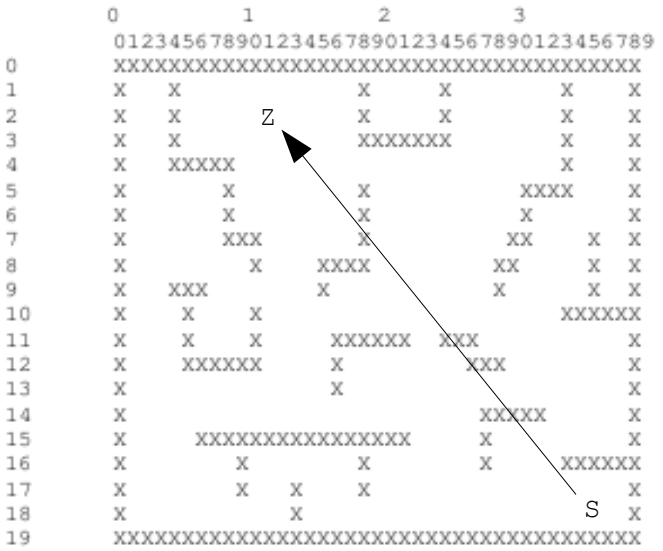
```
...
fib(5);
...
```



Aufgabe 1

a) Aufgabe verstehen

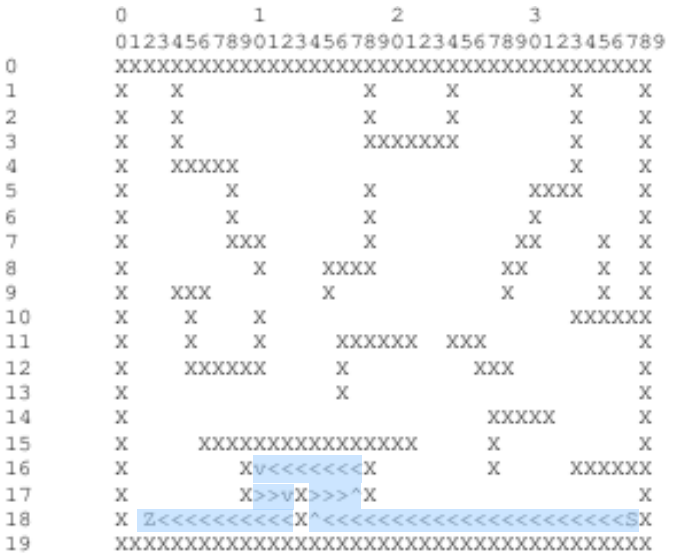
Ziel: Finde einen Weg von S (Start) nach Z (Ziel) in einem beliebigen Labyrinth:



Aufgabe 1

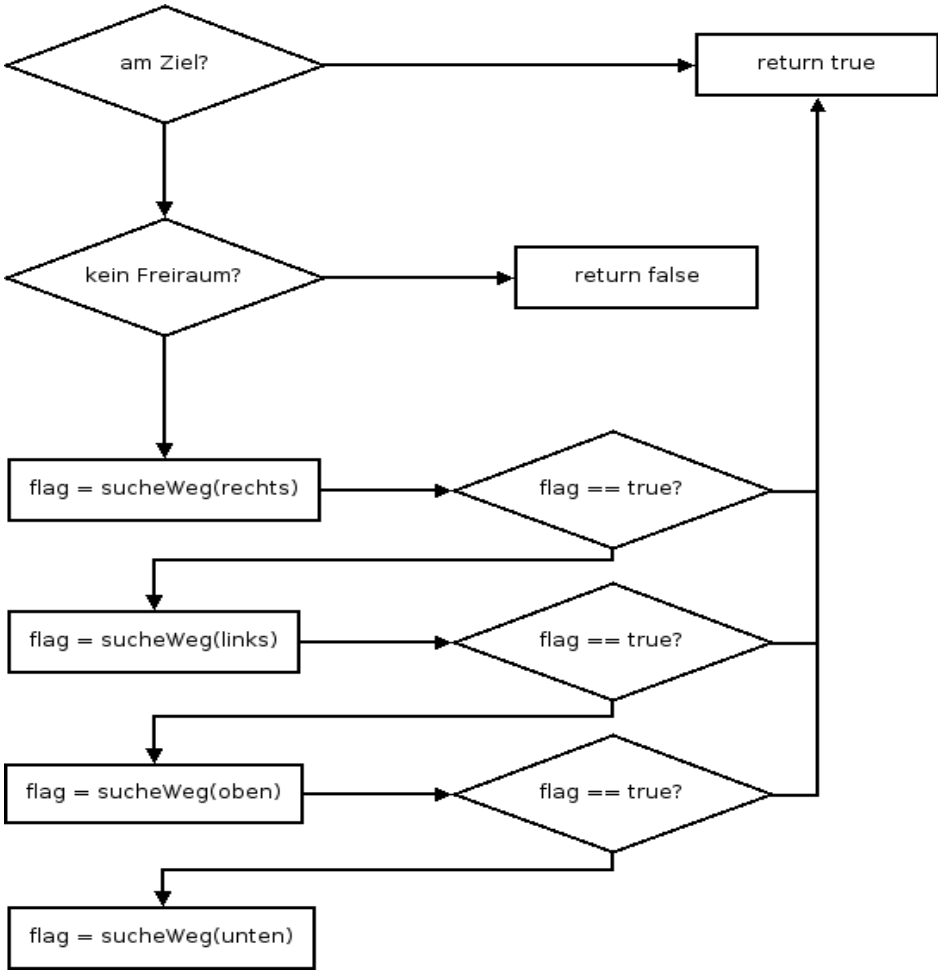
b) Algorithmus

Der Weg wird markiert im Feld:



Aufgabe 1

c) Rekursive Funktion

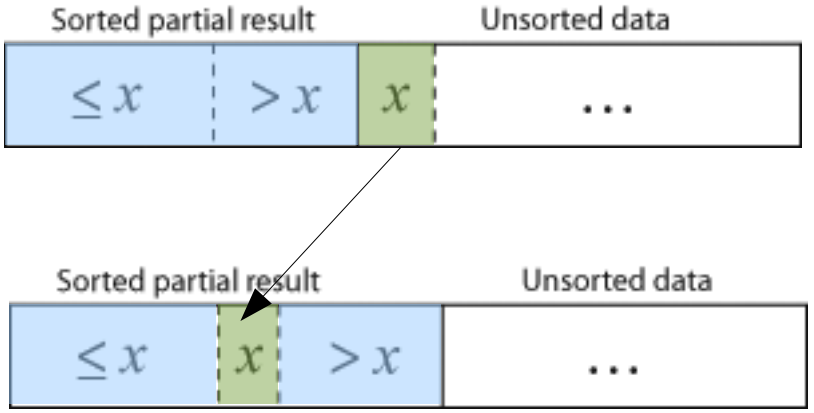


Insertion Sort

a) Begriffserklaerung

Liste wird **unterteilt** in **sortierten** und **unsortierten** Bereich

Elemente werden vom unsortierten Bereich in den sortierten Bereich eingefuegt (**insertion**)



Insertion Sort

a) Aufgabe

Schreibe eine Funktion

```
void sortierenDurchEinfuegen(int list[], int length)
{}
```

die eine Liste `<list>` (ist ein Array) mit der Laenge `<length>` sortiert mit dem insertion-sort Algorithmus

Informatik I

Nachbesprechung Uebung 7

Gruppe A, ETZ E91

Patrick Boenzli

dev.orxonox.net/wiki/boenzlip/inf1

Stack

a) Aufgabe verstehen

Schreibe einen **Stack** mit folgendem Interface:

`void` **`init()`** `{ ... }`
Initialisieren der Stack Daten

`int size()` `{ ... }`
Rueckgabe der Groesse des Stacks (Anzahl der Elemente)

`void` **`push(int element)`** `{ ... }`
Ablegen eines Elementes auf den Stack

`int pop()` `{ ... }`
Rueckgabe des obersten Elementes auf dem Stack

Stack

b) Loesung

Daten eines Stacks:

```
int* data;                  // Pointer auf das Array
int numberOfElements;       // Anzahl Elemente im Stack
int sizeOfArray;            // Kapazität des Arrays
```

Stack

b) Loesung

`void` **`init()`** `{ ... }`
Initialisieren der Stack Daten

```
void init()
{
  numberOfElements = 0;
  sizeOfArray = 2;
  data = new int[sizeOfArray];
}
```

Stack

b) Loesung

`int size()` `{ ... }`
Rueckgabe der Groesse des Stacks (Anzahl der Elemente)

```
int size()
{
  return numberOfElements;
}
```

Stack

b) Loesung

`void` **`push(int element)`** `{ ... }`
Ablegen eines Elementes auf den Stack

```
void push(int element)
{
  if ( sizeOfArray <= numberOfElements) {
      int* tempdata = new int[2 * sizeOfArray];
    for (int i = 0; i <= sizeOfArray; i++)
      {tempdata[i] = data[i];}
    sizeOfArray = 2* sizeOfArray;
    delete[] data;
    data = tempdata;
  }
  data[numberOfElements] = element;
  numberOfElements++;
}
```

Stack

b) Loesung

`int pop()` { ... }
Rueckgabe des obersten Elementes auf dem Stack

```
int pop()
{
  if (numberOfElements < 1){
    cout << "Stack underrun: Stack is empty!" << endl;
    return -1;
  } else {
    numberOfElements--;
    return data[numberOfElements];
  }
  return 0;
}
```

Stack

c) Allgemeine Fehler (dieser Code enthaelt 3 Fehler)

```
// globales array a
int* a = new int[1024];

// now fill array a and copy it to b
// ...
void someFunction() {
  int* b = new int[2048];
  for( int i = 0; i < 1024; i++)
    b[i] = a[i];
  delete [] a;

  a = b;
}
```

(von www.heise.de)