Rechnerpraktikum zu Kapitel 2

Objektorientierte Programmierung

C++-Standardbibliothek, Rekursion

Beispiel: Fast Fourier Transformation

Beispiel: Klasse für komplexe Zahlen

Aufgabe:

Definieren Sie im Hauptprogramm **main()** einen **vector<double>** und füllen Sie ihn mit 100 Zufallszahlen im Bereich von -10,0 bis +10,0.

Schreiben Sie eine Funktion **double find_max(vector<double> v)**, die der Reihe nach alle Elemente des Vektors durchläuft, das größte Element ermittelt und zurückgibt. Rufen Sie diese Funktion aus dem Hauptprogramm heraus auf und geben Sie das Ergebnis aus.

Tipps:

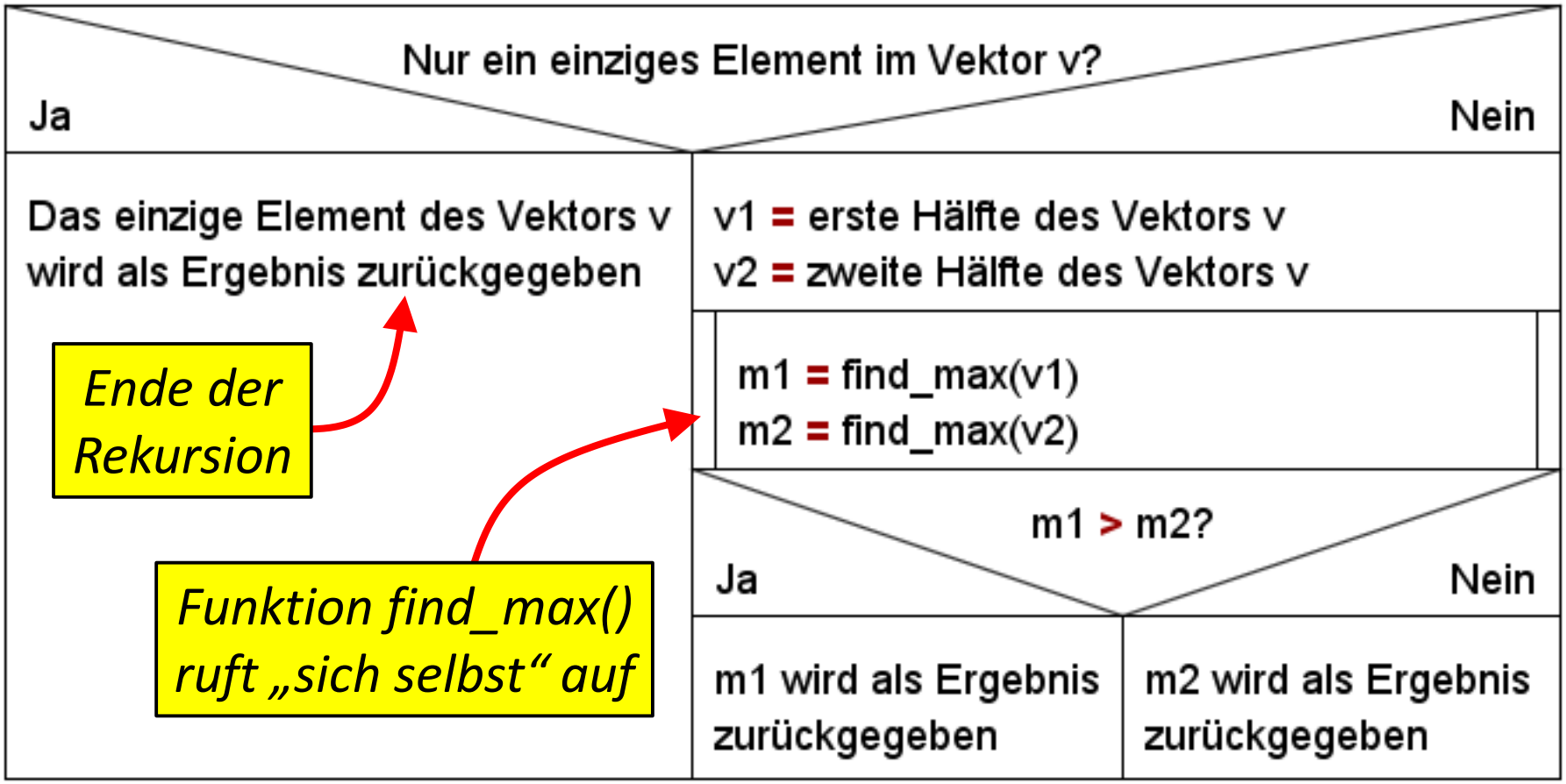
```
#include <vector>
using namespace std;

vector<double> v;
v.push_back(123.456);

cout << "Vektor-Elemente durchlaufen:" << endl;
for(auto x : v)
    cout << x << endl;
```

Aufgabe:

Implementieren Sie eine zweite Variante der Funktion **double find_max(vector<double> v)**, welche das Maximum rekursiv (!) ermittelt:



```
#include <iostream>
using namespace std;

double fakultaet(double x)
{
   if(x > 0)
     return fakultaet(x - 1) * x;
   else
     return 1;
}

int main()
{
   cout << fakultaet(3);
   cout << endl;
}
```

*Eine Funktion, die sich selbst aufruft, nennt man **rekursiv**. Das Rekursionsende muss definiert sein, damit keine „Endlos-Rekursion“ entsteht!*

fakultaet(0) { …. return 1; … }

fakultaet(1) { …. return fakultaet(0) * 1; … }

fakultaet(2) { …. return fakultaet(1) * 2; … }

fakultaet(3) { …. return fakultaet(2) * 3; … }

Ergebnis:

fakultaet(3) =

(((1) * 1) * 2) * 3 = 6

Rechnerpraktikum zu Kapitel 2

Objektorientierte Programmierung

C++-Standardbibliothek, Rekursion

Beispiel: Fast Fourier Transformation

Beispiel: Klasse für komplexe Zahlen

Was ist die Fast Fourier Transformation (FFT)?

Die FFT berechnet, vereinfacht ausgedrückt, welche Frequenzen im Originalsignal enthalten sind, welche Sinusschwingungen also mit welcher Phase addiert werden müssten, um auf das ursprüngliche Signal zu kommen.

Konkret erhält man für jeden Frequenzpunkt (von der Frequenz 0 bis zur halben Abtastfrequenz) eine komplexe Zahl, die Amplitude und Phase der entsprechenden Sinusschwingung repräsentiert.

(aus: www.mikrocontroller.net, Spektralanalyse mit der FFT)

Aufgabe:

Erstellen Sie eine Funktion **fft()** zur Berechnung der Fast Fourier Transformation eines Vektors aus komplexen Zahlen und testen Sie Ihre Funktion **fft()** mit einem geeigneten Hauptprogramm!

Struktogramm der Funktion fft(a):

Die Anzahl der Elemente muss eine Zweierpotenz sein!

n = Anzahl der Elemente im Vektor a

Anzahl n = 1?

Ja

Der Vektor a wird unverändert mit seinem einen Element als Ergebnis der Funktion fft() zurückgegeben.

Vergl.:

Introduction to Algorithms, Cormen et al., MIT Press, 2005

Nein

Komplexe Werte omega und omega_n berechnen:
omega = 1 und omega_n = exp(−2*pi*i/n), mit der imag. Einheit "i"

Zwei neue Vektoren a_i, a_ii erzeugen:
a_i = { a[0], a[2], a[4], ..., a[n-2] }
a_ii = { a[1], a[3], a[5], ..., a[n-1] }

Vektoren y_i und y_ii durch Selbstaufruf von fft() erzeugen:
y_i = fft(a_i) und y_ii = fft(a_ii)

Neuen Vektor y mit n komplexen Elementen anlegen und belegen:
Für alle Indizes k von 0 bis (n/2 - 1)...

```
y[k] = y_i[k] + omega * y_ii[k]
y[k + n/2] = y_i[k] - omega * y_ii[k]
omega = omega * omega_n
```

Vektor y als Ergebnis der Funktion fft() zurückgeben

```
#define _USE_MATH_DEFINES
#include <iostream>
#include <complex>
#include <vector>
#include <math.h>
#include "chart.h"
using namespace std;
typedef complex<double> cplx;
typedef vector<cplx> cplx_vec;
cplx_vec fft(cplx_vec a)
{
    int n = a.size();
    if(n == 1)
    {
        return a;
    }
    else
    {
        ...
```

*Dateien **chart-windows.c** und **chart.h** zum Projekt hinzufügen!*

***Zusatzaufgabe:** Wenn die Anzahl der Elemente im Vektor keine Zweierpotenz ist, soll die Funktion fft() mit einer Fehlermeldung abgebrochen werden!*

http://de.mathworks.com/help/matlab/ref/ff

Fast Fourier transform - MA...

MathWorks®

Documentation

CONTENTS Close

< All Products

< MATLAB

< Mathematics

< Fourier Analysis and Filtering

fft

ON THIS PAGE

Syntax

Definitions

Description

Examples

Data Type Support

More About

References

fft

Fast Fourier transform

Syntax

```
Y = fft(x)
Y = fft(X,n)
Y = fft(X,[],dim
Y = fft(X,n,dim)
```

Definitions

The functions `Y = fft(x)` and `y = ifft(X)` implement the transform and inverse

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

Vergleichen Sie die Ergebnisse Ihrer Funktion fft() mit den Ergebnissen der MATLAB-Funktion fft().

Tipp: In der MATLAB-Dokumentation zur Funktion fft() gibt es ein passendes Beispielprogramm!

Rechnerpraktikum zu Kapitel 2

Objektorientierte Programmierung

C++-Standardbibliothek, Rekursion

Beispiel: Fast Fourier Transformation

Beispiel: Klasse für komplexe Zahlen

P2.11. Klasse für komplexe Zahlen

Aufgabe:

Erstellen Sie ein neues C++-Projekt mit einer von Ihnen programmierten Variante der Klasse **Complex** zum Rechnen mit komplexen Zahlen. Ihre Klasse **Complex** sollte möglichst die folgenden Funktionalitäten bieten:

- Speichern, Abfragen von Real- und Imaginärteil
- Initialisieren neuer Instanzen durch geeignete Konstruktoren
- Operatoren für die vier Grundrechenarten
- Ermitteln von Betrag und Winkel
- Ausgabeoperator << für std::cout (und andere Ausgabe-Streams)
- Und noch eine Zusatzaufgabe für die Experten:
  Eingabeoperator >> für std::cin (und andere Eingabe-Streams)

P2.12. Klasse für komplexe Zahlen

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

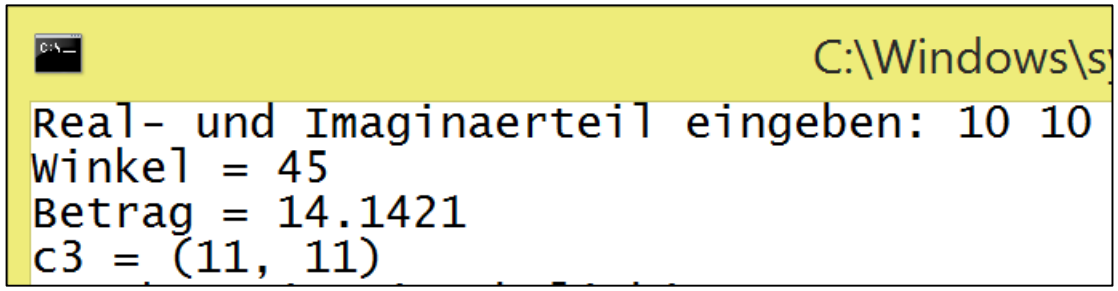
```
class Complex
{
    ...
};

int main()
{
    double r, i;
    cout << "Real- und Imaginaerteil eingeben: ";
    cin >> r >> i;

    Complex c1(r, i);
    cout << "Winkel = " << c1.Arg() * 180 / M_PI << endl;
    cout << "Betrag = " << c1.Abs() << endl;

    Complex c2;
    c2.Set(1, 1);

    Complex c3 = c1 + c2;
    cout << "c3 = " << c3 << endl;
}
```



So ähnlich könnte Ihr Hauptprogramm aussehen…