Projektarbeit Modul C++

Realisierung eines Algorithmus zur Anwendung des Heap-Sort in C++

Dozent: Dipl. Ing. Harald Sorber

Autoren:

Christian Piwecki, Patrick Schmid, Martin Sommer

Isny, den 24.02.2005

Inhaltsverzeichnis

Heapsort-Algorithmus

Theorie

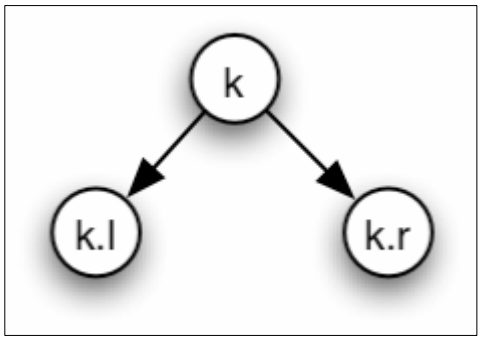
Einführung

J. Williams erfand 1964 einen neuen Weg eine Anzahl von Elementen zu sortieren. Anders als bisherige Sortierverfahren, die nach dem Minimum suchten, schlug er einen anderen Weg ein und ging von der Auswahl des Maximums aus. Er ordnete die Elemente zusätzlich in einen binären Baum ein, so dass für alle Knoten k mit Söhnen k.l, k.r gilt:

Bedingung an den Binärbaum

Binärbaum ist ein Heap, wenn der Schlüssel jedes Knotens mindestens so groß ist wie der Schlüssel seiner Nachfolger (falls es sie gibt).

Veranschaulichung:



$k \geq \max(k.l, k.r)$

Es muss also gelten:

$k \geq k.l$ und $k \geq k.r$, also $k \geq \max(k.l, k.r)$, soweit k.l und k.r existieren !
Diese sog Heap-Bedingung muss im ganzen Binärbaum gelten.

Somit ist die Wurzel w also das Element mit maximalem Schlüssel. Zur Bestimmung des zweitgrößten Elements benötigen wir nur einen einzigen Vergleich unter den beiden Söhnen der Wurzel.

Entwicklung

Am Anfang besteht der Heap aus Schlüsseln $k_1, \ldots k_n$, wobei n := Anzahl der Elemente im Baum.

Die folgenden 3 Stufen werden solange abgearbeitet, bis der Heap nur noch 1 Element enthält. Das letzte verbliebene Element bildet somit das kleinste Element in der Liste L und wird dieser noch angefügt.
Somit ist der Sortiervorgang abgeschlossen und man hat eine vollständig sortierte Liste vorliegen.

Stufe 1: „Wurzel ziehen“

Nehmen wir fortlaufend die Wurzel w aus dem Baum B heraus und fügen sie als neues

Minimum in eine anfangs leere Ergebnisliste L ein, so verletzen wir die Heap-Bedingung. B ist nicht einmal mehr ein Baum, da die Wurzel fehlt.
Es folgt die Wiederherstellung des Heap unter Berücksichtigung der Heap-Bedingung.

Stufe 2: „last -> first“

Williams nimmt ein Blatt des Baumes als neue Wurzel w. Er setzt also k_n (das Element an letzter Stelle des Baumes) auf die Stelle von k_1 (die erste „oberste“ Stelle des Baumes).

Stufe 3: „Sinken“

Die weitere Korrektur des Baumes besteht nun darin, die Heap-Bedingung durch den kompletten Baum laufen zu lassen:

$k.l < k$ und $k.r < k$.

Begonnen wird an der Wurzel, indem man jeweils die linken und rechten Nachfolger von k_1 betrachtet. Ist einer der Nachfolger größer als k_1, so tauscht k_1 den Platz mit dem größeren Nachfolger k_j. Ist das nicht der Fall verbleibt k_1 am Platz und Schleife wird beendet.
Der Tauschvorgang durch den Baum läuft nun solange (max. h-mal := Höhe des Baumes), bis beide Nachfolgerschlüssel nicht mehr größer als ihr Knoten sind,
also gilt:

$k_i \geq k_{2j}$ und $k_i \geq k_{2i+1}$, sofern $2i \leq n$ bzw. $2i+1 \leq n$ ist.

Erst dann ist die Heap-Bedingung wieder erfüllt.
Danach folgt wieder Stufe 1...

Bsp. für Stufe 3 des Algorithmus:

<C++ Code>

```
/*****************************************************************
                Die Baumausgleichsmethode
*****************************************************************/

void adjustheap(int arraysize, double daten[], int k){
  int anzahlel;
  double v;


 // Der Wert des gegebenfalls falsch platzierten Knotens mit Index K
 // wird dem Baum entnommen OHNE das Elemente aufrücken

  v = daten[k];

// Solange Knoten mit Index K mindestens einen Nachfolger hat wird fortgefahren

  while(k<arraysize/2){

// Berechnung der Anzahl der Elemente des Knotens K. Rechte Nachfolger ist durch
// Index anzahlel+1 gegeben

  anzahlel = 2*k+1;
// Wenn rechter Nachfolger vorhanden(anzahlel<arraysize-1) und der Wert grösseren
// Wert als linke Nachfolger (daten[anzahlel]<daten[anzahlel+1], so wird der
// rechte Nachfolger (anzahlel++) betrachtet
```

```
  if((anzahlel<arraysize-1) && (daten[anzahlel]<daten[anzahlel+1]))
      anzahlel++;
// Wenn Wert am Knoten K größer oder gleich Wert des Nachfolger ist kein weiteres
// Absteigen nötig und beenden der Schleife

  if(v>=daten[anzahlel])
      break;

// Nachfolger im Baum wird ein Level höher gezogen und Einstieg in Ast des
// Nachfolgers

  daten[k] = daten[anzahlel];
    k = anzahlel;
  }

// Gesicherter Wert des Störgliedes auf korrekte Position legen, die vom letzten
// aufrückenden Nachfolger frei gemacht wurde

  daten[k] = v;
}
```

</C++ Code>

Ergebnis der Entwicklung

Es kann max. nur n Durchläufe geben, da bei jedem Durchlauf die Wurzel herausgezogen wird. Beim letzten Element wird eigentlich nur noch die Wurzel in die Liste eingetragen, somit wären es effektiv sogar nur n-1 Durchläufe.

Praxis

vollständiger Sortieralgorithmus

<C++ Code>

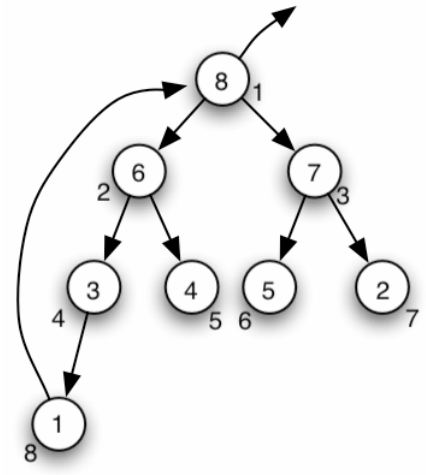
```
/*****************************************************************
                              Sortiermethode
*****************************************************************/

void heapsort(int arraysize, double daten[]){
  int k;
  double t;

// Beginn mit letztem Element im Baum, das Nachfolger hat(n/2).Drehen des
// Baumes an der Wurzel bis am Ende ein Heap entsteht.

  for(k=arraysize/2;k!=0;--k)
        adjustheap(arraysize, daten,k-1);

// Solange noch Elemente einzusortieren sind wird Schleifeninhalt ausgeführt

  while(--arraysize){

// Das grösste Element Daten[0] wird von Spitze des Heaps genommen und gegen
// das letzte noch zu betrachtende Element des Arrays ausgetauscht.
// Die Heap-Bedingung wird dadurch an daten[0] gestört

        t = daten[0];
        daten[0] = daten[arraysize];
        daten[arraysize] = t;

// In dem um ein Element verkleinerten Heap, an dem ggfs eine Störung an Wurzel
// (Index 0) vorliegt wird Reorganisation durchgeführt. Somit ist grösstes
// Element wieder an Spitze


        adjustheap(arraysize, daten, 0);
  }
}
```

</C++ Code>

Grafische Darstellung des Algorithmus mit Bäumen

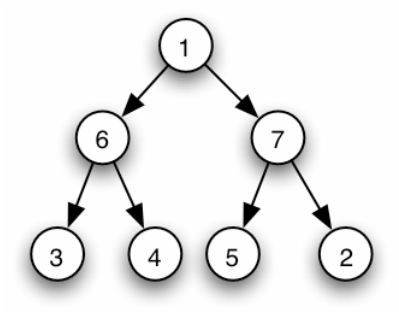
Folge F = 8, 6, 7, 3, 4, 5, 2, 1 erfüllt Heap-Bedingung

Folgender Baum wird aufgebaut:



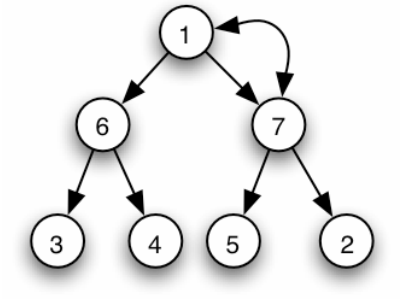
Heap als Binärbaum (8 fliegt raus)

Die weiteren Schritte bis zur vollständigen sortieren Folge:

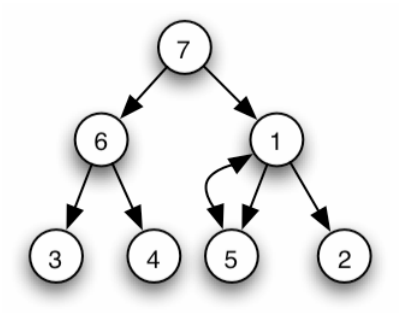


Heap Bedingung verletzt, Wurzel muss „versickern“

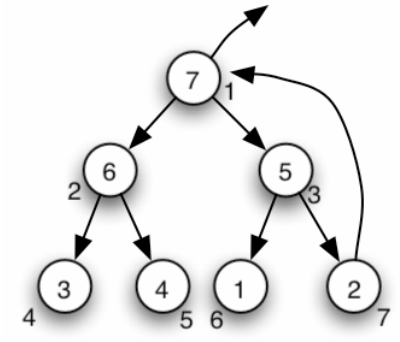
Tausch mit jeweils größtem Nachfolger



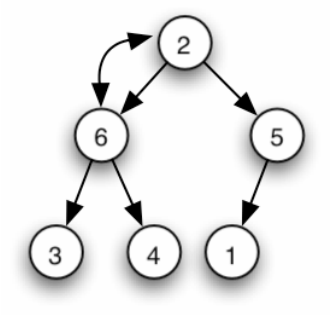
Tauschen von 1 und 7



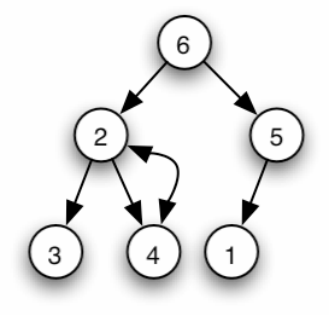
Tauschen von 1 und 5



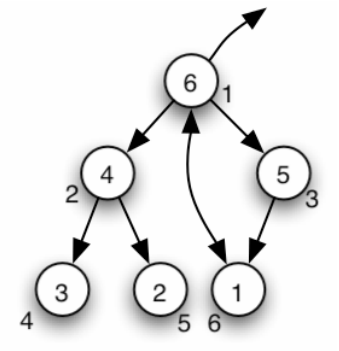
Heap-Bedingung wieder erfüllt ->7



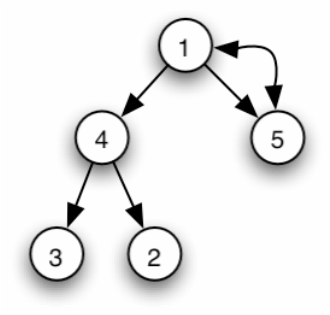
Tauschen von 2 und 6



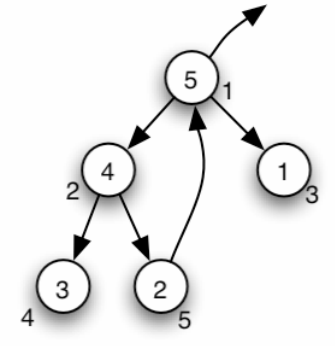
Tauschen von 2 und 4



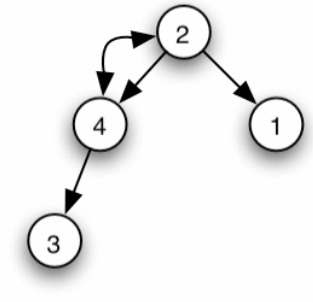
Heap-Bedingung erfüllt -> 6



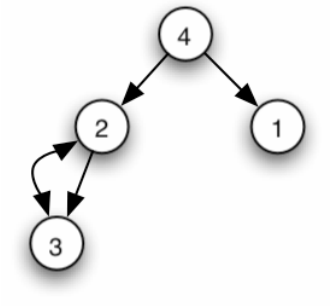
Tauschen von 1 und 5



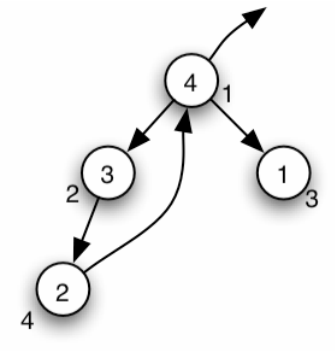
Heap-Bedingung erfüllt -> 5



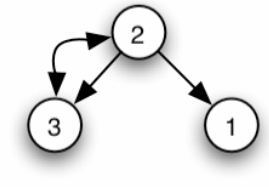
Tauschen von 2 und 4



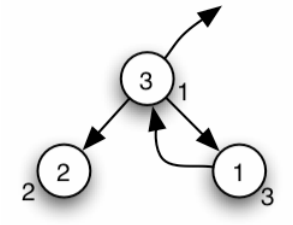
Tauschen von 2 und 3



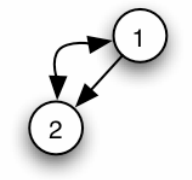
Heap-Bedingung erfüllt -> 4



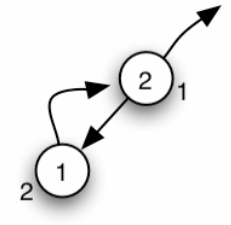
Tauschen von 2 und 3



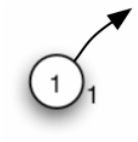
Heap-Bedingung erfüll -> 3



Tauschen von 1 und 2



Heap-Bedingung erfüllt -> 2



Heap-Bedingung erfüllt -> 1

Am Schluss erhält man als Ergebnis eine sortierte Folge von F = 8, 7, 6, 5, 4, 3, 2, 1

Zielsetzung/Programmbeschreibung

Unser Ziel war es, ein vor allem benutzerfreundliches Programm zu entwickeln, das den Heap-Sort Algorithmus verwirklicht. Gerade die Benutzerfreundlichkeit war uns sehr wichtig, da es oft passiert, dass Programme einfach abstürzen ohne dem Benutzer eine befriedigende Meldung auszugeben. Auch mangelnde Übersichtlichkeit oder fehlende Hilfetexte lassen den Benutzer öfters im Unklaren darüber, wie das Programm zu bedienen ist.
Wir machten uns also Gedanken darüber, wie wir dieses Programm so verbessern konnten, dass es einfach zu bedienen ist, aber auch bei Falscheingaben der Benutzer stets informiert, weshalb das Programm nicht weiter läuft.
Zu allererst stand natürlich das noch nicht fertige Design im Raum. Es musste was einfaches, dennoch gut aussehendes sein, was aber auch gleichzeitig wieder leicht zu bedienen war.
Wir kamen dann letztendlich auf die Idee, unser Programm in eine Art Menü zu verpacken, indem man mit einfachem Auswählen der Menüpunkte das Programm steuern kann.
Der Benutzer hat nun also die Möglichkeit zwischen den Menüpunkten

das Programm zu steuern.

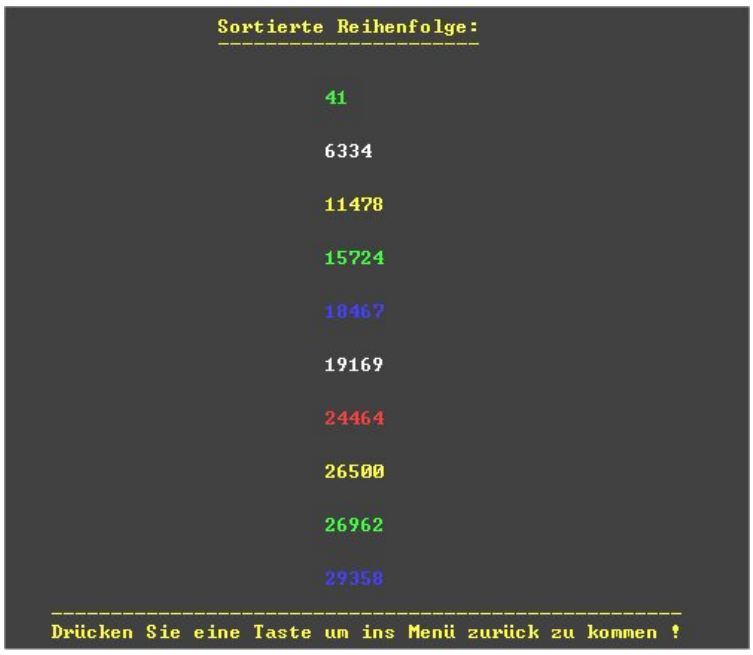
Gehen wir aber näher auf die verschiedenen Punkte ein.

In Punkt 1) **Heap-Sort mit eigener Eingabe**
Der Benutzer hat die Möglichkeit sich mit selbst ausgewählten Zahlen eine sortierte Reihenfolge anzeigen zu lassen. Jedoch kann es gerade in diesem Punkt sein, dass man Falscheingaben macht. Also überlegten wir uns eine passende Fehlerbehandlung, die den Benutzer so gut wie möglich über auftretende Programmabbrüche oder Verhaltensmuster informiert.

Wir realisierten es mit einem doch recht komplexem Exception Handling, welches jede Eingabe des Benutzers überprüft und im Puffer vergleicht, ob es eine Zahl ist oder nicht. So kann es also sein, dass bei Eingabe eines Buchstaben (anstatt einer Zahl) ein Fehlertext ausgegeben wird, der den Benutzer über eine mögliche Fehleingabe informiert.

So konnten wir sicher gehen, dass trotz falscher Angaben die Stabilität des Programms gewährleistet/nicht beeinflusst wird.

Kommen wir nun zu Punkt 2 – **Heap-Sort mit Zufallszahlen**
Dem Benutzer bietet sich die Möglichkeit, eine per Zufall generierte Zahlenreihenfolge von 10 Zahlen sortieren zu lassen. Wir realisierten dies mit der Funktion rand(), die einem erlaubt, Zufallszahlen zu generieren.



Der letzte Punkt unseres Programms Punkt 3) – **Beenden**
Er lässt den Benutzer, sofern er das überhaupt noch will, das Programm beenden.

Im Menü selbst, hat der Benutzer ja auch die Möglichkeit sich zu vertippen, sodass er anstelle einer 3 eine 4 eingibt, oder gar einen Buchstaben. Dies wurde von uns auch so realisiert, dass es praktisch gar nicht möglich ist, eine andere Eingabe als 1,2 oder 3 zu tätigen, sodass der Benutzer immer auf der sicheren Seite ist und noch viel Spaß an unserem Programm hat.

Probleme

Unser erstes Problem bestand darin den Heap zu programmieren. Das Einlesen der Daten bzw. Knoten funktionierte relativ schnell, doch das Sortieren stellte ein etwas größeres Problem dar. Er sortierte erst gar nicht, dann sortierte er falsch, dann nur die erste Zahl usw.
Wir kamen dann auf die Idee, dass wir zum Sortieren eine Vergleichsfunktion brauchten die uns aber gleichzeitig auch den Baum ausgleicht.
Im Quellcode die Methoden adjustheap() und heapsort().
Idealerweise macht dies eine while-Schleife…..

Wir hatten außerdem die Vorstellung, dass wir alle Benutzereingaben durch ein Exception Handling überprüfen wollten. Dazu zählten Eingaben von Buchstaben und Zeichen, anstelle von Ziffern und das Eingeben von nichts, also nur das Drücken der Enter Taste. Die ersten beiden Exceptions lösten wir mit if-Abfragen, die sich auf den Eingabe-Stream bezogen bzw. dessen was im Puffer stand. Also hatten wir etwas im Puffer, was wir vergleichen konnten.
Zweites Problem, also die Eingabe von nichts zu überprüfen stellte das größere Problem dar. Wir konnten also nicht irgendwas im Puffer auf etwas überprüfen, weil dort ja nichts beinhaltet war. Also überlegten wir uns eine Lösung mit Hilfe der Statusbits, die den Stream schon beim Einlesen überprüft und dann ein Fehlerbit setzt oder nicht.
Mit der Methode cin.fail() lassen wir nun eine Exception auswerfen, die auch für diese „leere Eingabe“ funktioniert.

Auswertung/Fazit

Unsere ersten Versuche basierten auf dem Binärbaum, was allerdings nicht wie gewünscht funktionierte. Eine Sortierung fand zwar statt, aber die Implementierung des Heap-Algorithmus gestaltete sich doch weit schwieriger als erwartet.
Deshalb übernahmen wird den Algorithmus nicht 1:1, sondern arbeiteten mit der Inorder-Traversierung wie gewöhnlich die Elemente des Baumes ab.
Mögliche Vorteile waren:

- Einfachere Arbeitsweise
- Keine Überprüfung der Heap-Bedingung mehr erforderlich
- Schnellere Abarbeitung/Steigerung der Effizienz
- Im Endeffekt ist das Ergebnis das gleiche

Damit wäre aber die Zielsetzung unseres Projekts nicht erreicht worden und zwar die Implementierung des Sortiervorgangs mit Hilfe von Heap-Sort.
Somit war eine weitere Recherche nötig.
Diese brachte eine neue Lösung ins Spiel. Meist wurde ein Array verwendet, um die Sortierung zu erreichen. Es reicht somit schon aus, die Elemente nur in ein Feld zu speichern und dann den Heap-Sort anzuwenden.
Also war unser nächster Schritt diesen neuen (alten) Ansatz aufzunehmen und damit unseren Algorithmus neu aufzubauen.
Das Zitat „Das Rad nicht neu erfinden“ war die neue These und drückt vielleicht am besten unsere Vorgehensweise aus.

Literatur

[1] Sorber, GDV2 Skript
[2] Goos, Band 2
[3] Kaiser, Kecher C++

Internetadressen

[4] http://www.sortieralgorithmen.de
[5] http://www.wikipedia.de

-Anhang-

Gesamter Quellcode

Unsere vorherigen gesammelten und aufgelisteten Ergebnisse haben wir nun in diesen Heapsort-Algorithmus für C++ eingearbeitet:

<C++ Code>

```
#include <iostream>
#include <sstream>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>
using namespace std;


// Definieren die Farben, welche später verwendet werden

#define green 2
#define blue  9
#define yellow 14
#define red 4
#define white 7
#define brightgreen 10
#define grey 8
#define black 0



// Methode die einem ermöglicht aufgrund von definierten Farbnamen diese zu
// verwenden mit Eingabe des Farbwortes

void textcolor(WORD color)
        {
                SetConsoleTextAttribute(::GetStdHandle(STD_OUTPUT_HANDLE), color);
        }




/*****************************************************************
                Die Baumausgleichsmethode
*****************************************************************/

void adjustheap(int arraysize, double daten[], int k){
  int anzahlel;
  double v;
```

```
 // Der Wert des gegebenfalls falsch platzierten Knotens mit Index K
 // wird dem Baum entnommen OHNE das Elemente aufrücken

  v = daten[k];

// Solange Knoten mit Index K mindestens einen Nachfolger hat wird fortgefahren

  while(k<arraysize/2){

// Berechnung der Anzahl der Elemente des Knotens K. Rechte Nachfolger ist durch
// Index anzahlel+1 gegeben

          anzahlel = 2*k+1;
// Wenn rechter Nachfolger vorhanden(anzahlel<arraysize-1) und der Wert grösseren
// Wert als linke Nachfolger (daten[anzahlel]<daten[anzahlel+1], so wird der
// rechte Nachfolger (anzahlel++) betrachtet

    if((anzahlel<arraysize-1) && (daten[anzahlel]<daten[anzahlel+1]))
      anzahlel++;
// Wenn Wert am Knoten K größer oder gleich Wert des Nachfolger ist kein weiteres
// Absteigen nötig und beenden der Schleife

          if(v>=daten[anzahlel])
      break;

// Nachfolger im Baum wird ein Level höher gezogen und Einstieg in Ast des
// Nachfolgers

          daten[k] = daten[anzahlel];
    k = anzahlel;
  }

// Gesicherter Wert des Störgliedes auf korrekte Position legen, die vom letzten
// aufrückenden Nachfolger frei gemacht wurde

  daten[k] = v;
}


/*****************************************************************
                                                Sortiermethode
*****************************************************************/

void heapsort(int arraysize, double daten[]){
  int k;
  double t;

// Beginn mit letztem Element im Baum, das Nachfolger hat(n/2).Drehen des
// Baumes an der Wurzel bis am Ende ein Heap entsteht.

  for(k=arraysize/2;k!=0;--k)
  adjustheap(arraysize, daten,k-1);

// Solange noch Elemente einzusortieren sind wird Schleifeninhalt ausgeführt
```

```
  while(--arraysize){

// Das grösste Element Daten[0] wird von Spitze des Heaps genommen und gegen
// das letzte noch zu betrachtende Element des Arrays ausgetauscht.
// Die Heap-Bedingung wird dadurch an daten[0] gestört

        t = daten[0];
    daten[0] = daten[arraysize];
    daten[arraysize] = t;

// In dem um ein Element verkleinerten Heap,an dem ggfs eine Störung an Wurzel
// (Index 0) vorliegt wird Reorganisation durchgeführt. Somit ist grösstes
// Element wieder an Spitze


        adjustheap(arraysize, daten, 0);
  }
}


/*****************************************************************
Methode für Exception Handling. Überprüft ob Eingabe zulässig war.
*****************************************************************/

/*
* String einlesen
* Exception Handling
* Testen auf zulaessige Werte mit Fehlerbehandlung
*/


class DateiEnde{}; // Hilfsklasse Exception

double fromString (string &blub, double &i) // testet auf Double
{
        istringstream converter (blub); // converter ist der Puffer

    converter >> i;             // Puffer kommt in die Eingabe -> double

        if (converter.fail())          // sind Fehlerflags gesetzt ?
        {
                throw
                        "                    Unzulaessige Eingabe !"; // es ist keine Zahl (z.B.
asdfsa, sadsa56, asd 342, sdfd sdf, ...)
        }
        converter >> ws; // vorhergehende Leerzeichen werden entfernt, eof wird gesetzt

        if (!converter.eof()) // Ende des Streams noch nicht erreicht ?
        {
                throw
                        1; // es ist eine Zahl, aber mit Anhaengsel (z.B. 3asfda, 2 dsfsd, ...)
        }
```

```
        if (converter.bad()) // nicht definiert
        {
                throw
                        DateiEnde(); // nicht behebbbarer Fehler
        }

        return i; // ansonsten alles in Ordnung
}



double Eingabe()
{
        string eingabe;


        for (;;)
        {
           bool erfolgreich = true;
                double zahl;


       getline (cin, eingabe); // als String einlesen
                //cin >> eingabe;

                try // Versuchsblock
                {
                        zahl = fromString (eingabe, zahl);
                }

                catch (DateiEnde) // Fehlerbehandlung, definierter Abbruch
                {   textcolor(red);
                        cerr << "                    Ende" << endl;

                        textcolor(white);
                        cout<<"                    ";

          cin.clear();



                }

                catch (int) // Fehlerbehandlung bei double, float, +Anhaengsel
                {   textcolor(red);
                        cerr << "                    Keine Zahl!" << endl;

                        textcolor(white);
                        cout<<"                    ";


          erfolgreich = false;
```

```
                cin.clear();
                //cin.get();
        }

        catch (const char *z) // Fehlerbhandlung bei Zeichen, +Anhaengsel
        {
                textcolor(red);
          cerr << z << endl;

                textcolor(white);
                cout<<"                    ";

                erfolgreich = false;

                //cin.get();
          cin.clear();



    }

        if (erfolgreich) // Eingabe war Integer
        {
                return zahl;
        }
     }
}


/*********************************************************************
    Algorithmus zum Sortieren in einem Binärbaum.
*********************************************************************/


struct node{
   double data;
   node* r;
   node* l;
};

class nodeHandle{
   node* start;
   node* ende;



   public:
   nodeHandle(){
      start = new node;
      ende = new node;
      start->l = ende;
      start->r = ende;
      ende->l = ende;
      ende->r = ende;
```

```
    }

    void addNode(double data);
    void show();
    void inorder(node* x);
    //void getTree(double* darray[]);
};

void nodeHandle::addNode(double data){
    node* newnode = new node;
    newnode->data = data;
    newnode->l = ende;
    newnode->r = ende;

    if(start->r==ende){
        start->r = newnode;
    }
    else{
        node* current = start->r;
        node* prev = start;

        while(current!=ende){
                prev = current;
                current->data < data ? current = current->r : current = current->l;
        }


    current = newnode;

    if(data <= prev->data)              prev->l = current;
    else                            prev->r = current;
    }
}


void nodeHandle::inorder(node* x){
    char color[] = {green, white, yellow, brightgreen, blue, grey, red, yellow, brightgreen,
blue};


    if(x!=ende){
        inorder(x->l);


        textcolor(color[4]);

                                    cout <<"\n\n                                "<< x->data << endl;
                                    cout<<endl;


        inorder(x->r);
    }
}
```

```
void nodeHandle::show(){
   textcolor(yellow);
                cout<<endl<<endl<<endl<<endl;
                cout<<"                        Sortierte Reihenfolge:"<<endl;
      cout<<"                        ----------------------"<<endl;

   inorder(start->r);

   textcolor(yellow);
                cout<<"           ------------------------------------------------------"<<endl;
                cout<<"           Dr"<<char(129)<<"cken Sie eine Taste um ins
Men"<<char(129)<<" zur"<<char(129)<<"ck zu kommen !"<<endl;
      textcolor(white);
}

void treeSort(double darray[],int size){
   // Baum erstellen
   nodeHandle Tree1;

   for(int i=0; i<size; i++){
      Tree1.addNode(darray[i]);       // Element zum Baum hinzufügen
   }

   system("cls");
   Tree1.show();                    // sortieren Baum anzeigen
   getch();
}




int main()
{ double darray[10];
  double darray2[10];
  int size;
  char choice;
  bool erfolgreich;
  bool standardsort;

  double zahl;
  int i;
  do{
      system("cls");
           //clrscr();

// Legt vordefinierten Farbwert fest

                textcolor(red);
                cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;
                cout<<"
*************************************************************************"<<endl;
                textcolor(grey);
```

```
                cout<<"     |                       Auswahlmen"<<char(129)<<"
|"<<endl;
        textcolor(red);
                cout<<"
**********************************************************************"<<endl;
                cout<<"                             *         *                    "<<endl;
                textcolor(grey);
                cout<<"                               Heap-Sort"<<endl;
                textcolor(red);
                cout<<"                             *         *          "<<endl;
                cout<<"     ----------------------------------------------------------------------"<<endl;
                cout<<"     |                                                      |"<<endl;
                cout<<"     |                                                      |"<<endl;
                //textcolor(yellow);
                cout<<"                       1) Heap-Sort mit eigener Eingabe"<<endl;
        textcolor(red);
                cout<<"     |                                                      |"<<endl;
                cout<<"     ----------------------------------------------------------------------"<<endl;
                cout<<"     |                                                      |"<<endl;
                cout<<"     |                                                      |"<<endl;

                //textcolor(blue);
                cout<<"                       2) Heap-Sort mit Zufallszahlen"<<endl;
        textcolor(red);
        cout<<"     |                                                 |"<<endl;
                cout<<"     ----------------------------------------------------------------------"<<endl;
                cout<<"     |                                                      |"<<endl;
                cout<<"     |                                                      |"<<endl;

                //textcolor(green);
                cout<<"                       3) Heap-Sort(Baum) mit eigener Eingabe
"<<endl;
        textcolor(red);
                cout<<"     |                                                      |"<<endl;
                cout<<"     ----------------------------------------------------------------------"<<endl;
          cout<<"     |                                                |"<<endl;
                cout<<"     |                                                      |"<<endl;

                //textcolor(green);
                cout<<"                       4) Heap-Sort(Baum) mit Zufallszahlen
"<<endl;
        textcolor(red);
                cout<<"     |                                                      |"<<endl;
                cout<<"     ----------------------------------------------------------------------"<<endl;
        cout<<"     |                                                 |"<<endl;
                cout<<"     |                                                      |"<<endl;

                //textcolor(green);
                cout<<"                       5) Beenden                "<<endl;
        textcolor(red);
                cout<<"     |                                                      |"<<endl;
                cout<<"     ----------------------------------------------------------------------"<<endl;
                textcolor(white);
```

```
        cout<<endl;
        cout<<"     Ihre Auswahl: ";


         while(true){

                choice = getch();
                if(choice=='1'||choice=='2'||choice=='3'||choice=='4'||choice=='5')break;
           }


         switch (choice)
          {  case '1':    standardsort = true;
                        system("cls");


cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;
                                        textcolor(yellow);
                                                cout<<"                 Geben Sie 10 Zahlen nach
dem Piepton ein"<<endl;
                        cout<<endl<<endl;
                                                textcolor(red);
                                                cout<<"                 3..."<<endl;

// Sleep Funktionen zum Timen des Signals

                                                Sleep(1000);
                        textcolor(blue);
                                                cout<<endl<<endl;
                                                cout<<"                 2..."<<endl;
                        Sleep(1000);
                        textcolor(green);
                                                cout<<endl<<endl;
                                                cout<<"                 1..."<<endl;
                        Sleep(1000);
                        cout<<"\a"<<"\a"<<"\a";
                        textcolor(white);
                                                system("cls");
                                                //clrscr();

// Eingabeschleife von 10 Zahlen

                                                for (i=0;i<10;i++)
                        {   cout<<endl<<endl<<endl;
                                                        textcolor(white);
                            cout<<"                 Zahl"<<i+1<<": ";


                            double x = Eingabe();
                            darray[i] = x; //Übergabe der Eingabe an das Array
                        }

                        break;
```

```
        case '2':    standardsort = true;

                     for(i=0;i<10;i++)

                                                  // Funktion rand() erzeugt Zahlen aufgrund der
Zeit der verganengen Sekunden
                                                  // seit Anfang der Epoche also 1.1.1970

                                                    darray[i]=rand();
                     break;


        case '3':    standardsort = false;
                     system("cls");

                     for (i=0;i<10;i++)
                     {   cout<<endl<<endl<<endl;
                                                        textcolor(white);
                        cout<<"               Zahl"<<i+1<<": ";


                        double x = Eingabe();
                        darray2[i] = x; //Übergabe der Eingabe an das Array
                     }


                     size = sizeof(darray2)/sizeof(double);
                     treeSort(darray2,size);

                     break;


        case '4':    standardsort = false;
                     system("cls");

                     for (i=0;i<10;i++)
                     {   cout<<endl<<endl<<endl;
                                                        textcolor(white);
                        cout<<"               Zahl"<<i+1<<": ";


                        darray2[i] = rand(); //Zufallszahl an das Array übergeben
                     }


                     size = sizeof(darray2)/sizeof(double);
                     treeSort(darray2,size);

                     break;



        case '5':    system("cls");
                                                  //clrscr();
```

```
cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<
endl;
                                        textcolor(yellow);
                                                cout<<"             Vielen Dank f"<<char(129)<<"r
die Benutzung dieses Programms"<<endl;
                        cout<<"\a";
                        cout<<endl<<endl<<endl<<endl<<endl<<endl;
                                                textcolor(grey);
                                                cout<<"     (C) 2005 < Martin Sommer > < Patrick
Schmid > < Christian Piwecki >"<<endl;
                                                getch();
                        exit(0);

                default:    cout<<endl;
                        textcolor(yellow);
                        cout<<"\n     Error#239 Exception on Address 676A7872x8234 oder
falsche Eingabe";
                        getch();
                }

// Methodenaufruf zum Sortieren der Zahlen mit Standardalgorithmus
if(standardsort){
                heapsort(10,darray);

                system("cls");
                //clrscr();

// Ausgabe der sortierten Zahlenreihenfolge
                textcolor(yellow);
                cout<<endl<<endl<<endl<<endl<<endl;
                cout<<"                         Sortierte Reihenfolge:"<<endl;
        cout<<"                          ----------------------"<<endl;
                cout<<endl;
                textcolor(white);
        char color[] = {green, white, yellow, brightgreen, blue, grey, red, yellow, brightgreen,
blue};

        for (int i = 0; i < 10; i++)
        {
                textcolor(color[i]);
                cout<<endl;
                                cout <<"                                "<< darray[i] << endl;
                                cout<<endl;
        }



                textcolor(yellow);
                cout<<"           ---------------------------------------------------"<<endl;
                cout<<"           Dr"<<char(129)<<"cken Sie eine Taste um ins
Men"<<char(129)<<" zur"<<char(129)<<"ck zu kommen !"<<endl;
        textcolor(white);
                getch();
```

```
}

    }while(choice!=3);

getch();

return 0;
}

</C++ Code>
```