
Name (Blockschrift) Matrikelnummer

Hinweise:
Bearbeitungszeit: 120 Minuten
Hilfsmittel: Ein nichtprogrammierbarer Taschenrechner.

- Bearbeiten Sie die Aufgaben direkt und nur auf den Aufgabenblättern oder Leerbögen.
- Falls der vorgesehene Platz zur Lösung der Aufgabe nicht ausreicht (bitte auch die Folgeseiten auf Leerseiten prüfen), verwenden Sie bitte die Leerseiten zur Fortsetzung ihrer Lösung. Dies ist bei der Aufgabenstellung mit ….**weiter auf Seite xx** … zu dokumentieren!
- Lösungen ohne erkennbaren und leserlichen Lösungsweg sind wertlos.
- Klausur zusammenlassen!!!
- Deckblatt mit Namen und Matrikelnummer beschriften und unterschreiben!
- Die letzte Seite der Klausur beinhaltet eine STL-Hilfe.

Hiermit bestätige ich, dass ich die Hinweise zur Kenntnis genommen habe:

(Unterschrift)

Aufgabe	Max. Pkt.	Punkte	
1	**26**		
2	**11**		
3	**18**		
4	**11**		
5	**10**		
Gesamt	**76**		**Note:**

Aufgabe 1 (26 Punkte)

(a) Ein Knoten eines Baumes kann einen Zahlenwert vom Datentyp Integer speichern und referenziert auf N Nachfolgeknoten. Geben Sie dazu eine passende Datenstruktur in C++ an und implementieren Sie einen Konstruktor, der als Eingabe die Knotenanzahl N übergeben bekommt und den Speicher allokiert. (4P)

```
class Node{
  public: int item; std::vector<Node*> nachfolger;
          Node(int N){item=0; nachfolger.resize(N, nullptr);}
};
```

(b) Nennen Sie eine geeignete Datenstruktur, um gewichtete Kanten in einem Graphen aufsteigend sortiert abzuarbeiten. (1P)

Priority Queue

(c) Nennen Sie eine Komplexitätsklasse, bei denen eine Berechnung zu aufwändig wird. Geben Sie ein Beispiel aus der Vorlesung dazu an. (2P)

$O(n^2)$ → BubbleSort

(d) Ein Array aus Integerzahlen soll mit der STL mit den folgenden Werten initialisiert werden: 10, 9, 8. Geben Sie den notwendigen C++ Code dazu an. (3P)

```
vector<int> a{10,9,8};
```

(e) Nennen Sie zwei Kriterien, die die Effizienz eines Algorithmus bewertet. (2P)

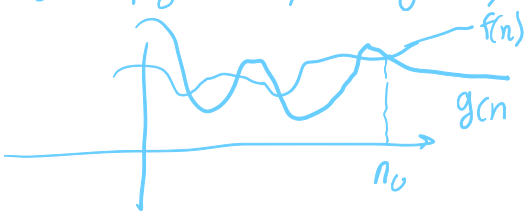
Time and Space Complexity

(f) Definieren Sie die O-Notation. Was ist der „Cross Over Point“? Und was kennzeichnet den “Cross Over Point” in der O-Notation? Erläutern Sie dies am besten mit einer Skizze. (3P)

$g(n) = O(f(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} \geq c : c > 0$

n_0 ist CrossOverPoint zwischen $O(g(n))$ und $O(f(n))$

$\Leftrightarrow \forall n \quad n > n_0 \Rightarrow g(n) < f(n)$



(g) Welche Traversierungsmethode besucht die Knoten innerhalb eines binären Suchbaums entlang eines Pfades vom Blatt bis zur Wurzel? (1P)

TiefenSuche

(h) Welche Datenstrukturen kennen Sie für ausgeglichene Bäume? (2P)

B+-B Bäume AVL 234-Baum Rot schwarz tree

(i) Ein Stapel ist als einfach verkettete Liste implementiert, der Kopfzeiger sei Node* head. Das erste Datenelement sei head->next. Schreiben Sie eine C++-Methode, um in einen Stapel ein Element in O(1) hinzuzufügen. (3P)

```
void stapel::add(Node* newNode)
{
    newNode->next = head->next;
    head->next = newNode;
}
```

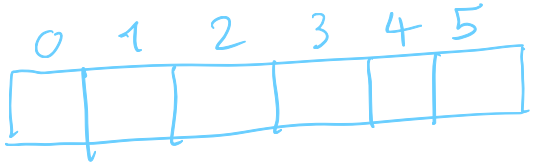
(j) Nach welchem Prinzip arbeitet eine Queue? (1P)

FIFO First in First out

(k) Ein Ringpuffer soll mit der Datenstruktur Array implementiert werden. Der Puffer soll 5 Integer-Elemente vom Anwender aufnehmen können.

1. Skizzieren Sie den entsprechenden Ringpuffer. (1P)
2. Implementieren Sie in C++ den Konstruktor, der den notwendigen Speicher allokiert (ohne STL) und die für den Ringpuffer notwendigen Variablen zum Lesen und Schreiben initialisiert. (3P)

1.

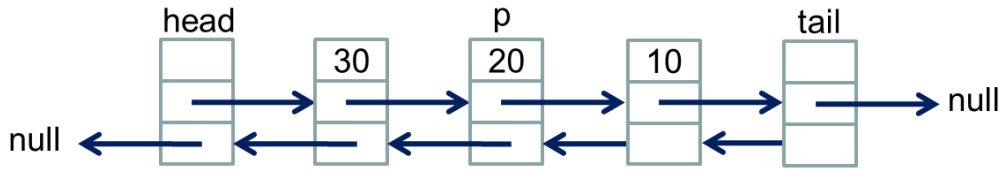



```
Puffer(){
    arr = new int(6);
    head = 0;
    tail = 5;
}
```

Aufgabe 2 (11 Punkte)
Methoden in einer doppelt verketteten Liste (ohne STL)

Gegeben sei eine doppelt verkettete Liste, dynamisch programmiert, deren Elemente absteigend sortiert eingefügt wurden. Ein Knoten hat die folgende Klassendefinition:

```
class Node {
public:
   int item;
   Node* next;
   Node* prev;
   Node() { item = 0; next = nullptr; prev = nullptr;}
};
```



In der Datenstruktur List markieren „head“ und „tail“ den Anfang bzw. das Ende der Liste und sind selber **nicht** mit Daten belegt.

(a) Schreiben Sie eine C++-Methode, die einen gegebenen Knoten p aus der Liste löscht, dh. der Knoten muss nicht in der Liste gesucht werden. Die Adresse des zu löschenden Knoten sei bekannt und wird der Methode als Parameter übergeben. (3P)

```
void List::deleteNode(Node* p)
```

(b) Schreiben Sie eine Methode, um 2 Knoten p1 und p2 zu vertauschen. Beachten Sie auch den Fall, dass die zu vertauschenden Knoten in der Liste direkt hintereinander liegen können. (8P)

```
void List::swapNodes(Node* p1, Node* p2)
```

a)
```
void List::deleteNode(Node* p)
{   if(!p || p==head || p==tail)
        return;
    p->prev->next = p->next;
    p->next->prev = p->prev;
    delete p;
}
```

b)

```
Void List::swapNodes(Node *p1, Node *p2)
{   if(!p1 || !p2)
        return;

    bool hintereinander = (p1->next == p2) || (p1->prev == p2);

    Node *erste = (p2->next == p1) ? p2 : p1;
    Node* zweite = (erste == p1) p2 : p1;

    erste -> prev -> next = zweite;
    zweite -> next -> prev = erste;

    Node * tmp; // für swap

    if(hintereinander) {
        erst->next = zweite->next;
        zweite->next = erste;
        zweite->prev = erste->prev;
        erste->prev = zweite;

    } else {
        erste->next -> prev = zweite;
        zweite -> prev -> next = erste;
            tmp = erste->prev;
            erst->prev = zweite->prev;
            zweite->prev = tmp;

            tmp = erste->next;
            erste->next = zweite->next;
            zweite->next = tmp;
    }
}
```

man kann von hier anfangen, wenn man vermutet, dass p1 vor p2 in die Liste kommt.

Aufgabe 3 (18 Punkte)
Bäume

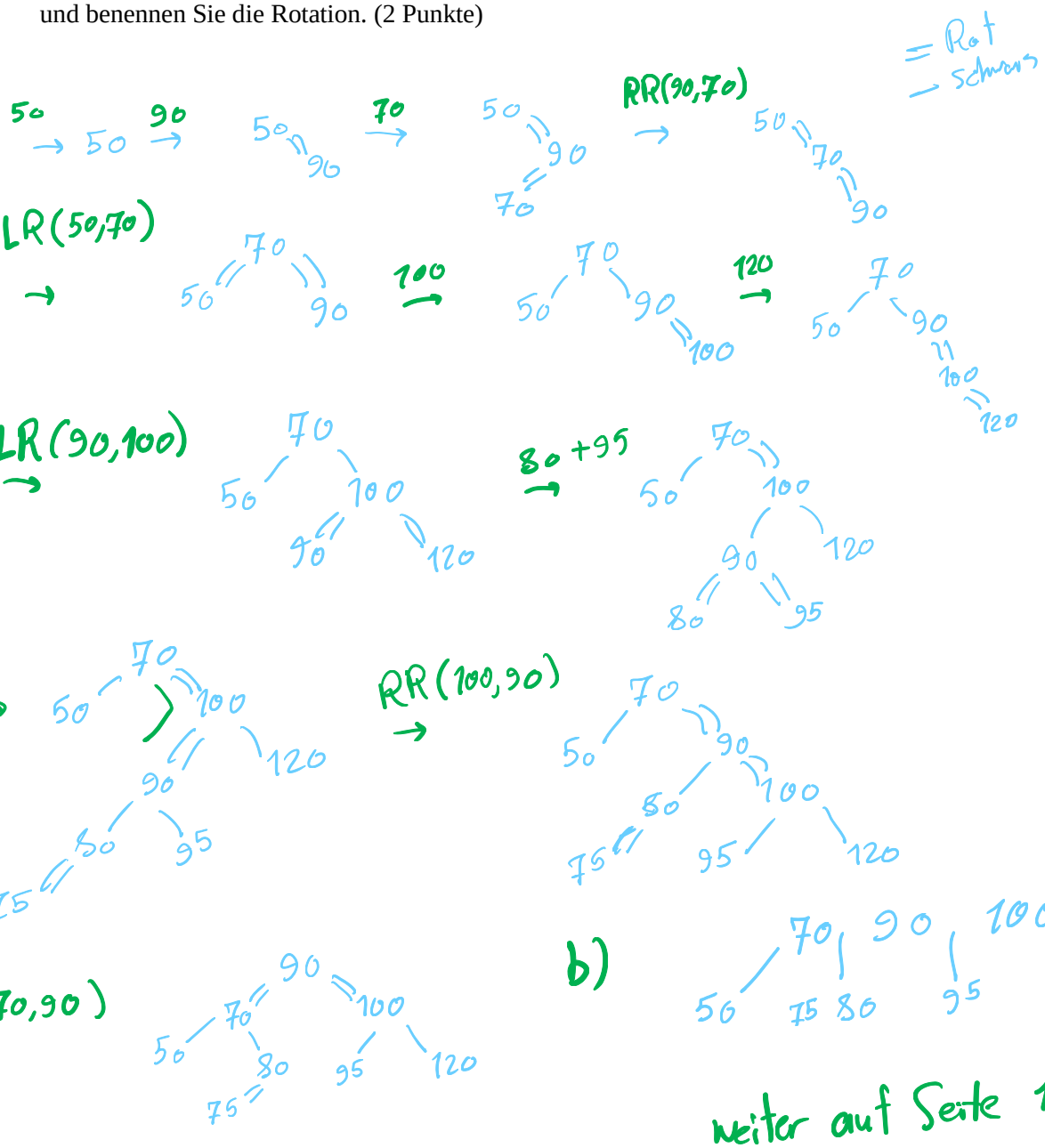
Sie sollen in einen Baum Elemente vom Datentyp Integer einfügen.

(a) Fügen Sie nacheinander in einen anfangs leeren Rot-Schwarz-Baum die folgenden Elemente ein: 50, 90, 70, 100, 120, 80, 95, 75. Benennen Sie die Rotationen und führen diese sichtbar durch. (5 Punkte)

(b) Wie lautet der zugehörige 234-Baum zu ihrer Lösung aus (b)? (2 Punkte)

(c) Skizzieren Sie mit den Werten 10, 20, 30 die Fälle für einen Rot-Schwarz-Baum, in denen zwei rote Knoten aufeinanderfolgen. Skizzieren Sie auch die Lösung, die durch die Rotationen entsteht und benennen Sie die Rotation. (2 Punkte)

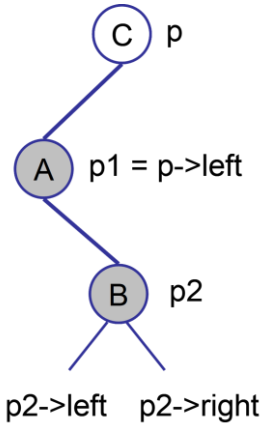
a)



b)

weiter auf Seite 13

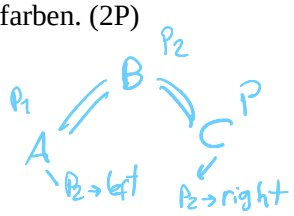
(d) Implementieren Sie eine Links-Rechtsrotation in einem Rot-Schwarz-Baum ohne zwei einzelne Rotationen zu verwenden.



Knoten C ist schwarz, die Knoten A und B sind rot. Die Funktion soll die Linksrotation zwischen A und B und dann anschließend die Rechtsrotation zwischen C und B durchführen. Überlegen Sie, ob und wie Sie die Farben der Knoten nach der Rotation anpassen müssen.

```
class treenode {
   public:
      int item;
      treenode *left;
      treenode *right;
      bool color; // true=rot
}
```

- Skizzieren Sie zunächst das Ergebnis der Links-Rechtsrotation mit allen Knoten, Referenzen und Knotenfarben. (2P)



- Implementieren Sie nun die Links-Rechts-Rotation zusammen in einer Methode (7P):

void RSTree::LeftRightRotation(treenode* &p)

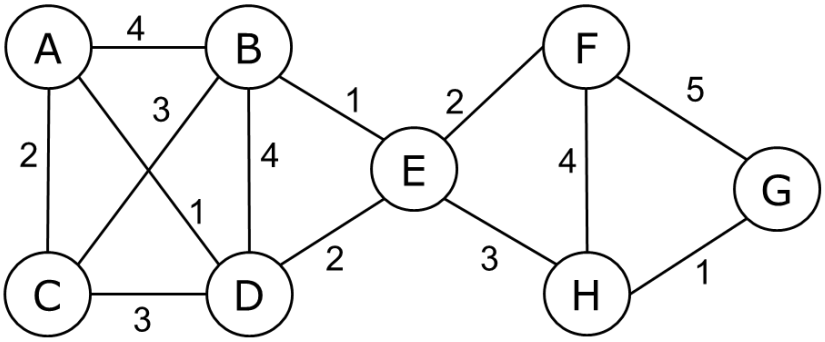
```
{   if(!p || !p->left || !p->left->right)
        return;
    treenode *p1 = p->left, *p2 = p1->right;
    p1->right = p2->left;
    p->left = p2->right;
    p2->left = p1;
    p2->right = p;
    p2->color = 0;
    p->color = 1;
}
```

Aufgabe 4 (11 Punkte)

Graphen

(a) Gegeben sei ein Graph mit 8 Knoten. Wie viele Kanten hätte der Graph, wenn jeder Knoten mit jedem anderen verbunden wäre (außer mit sich selbst)? Wie nennt man einen solchen Graphen? (2P)

Gegeben sei nun der folgende Graph:



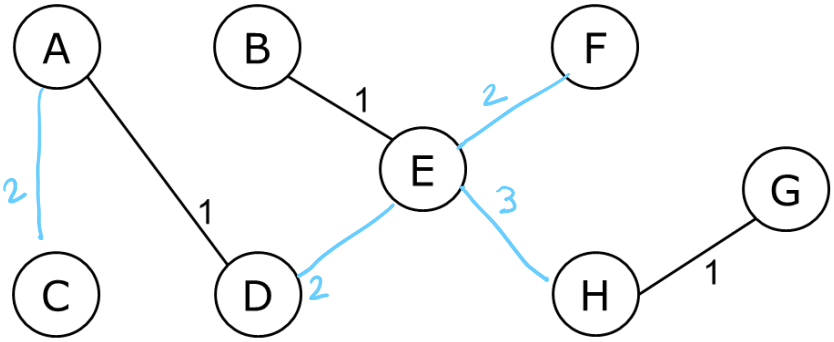
(b) Wie lautet die Adjazenzmatrix zu dem Graphen? (2P)

	A	B	C	D	E	F	G	H
A	0	4	2	1	0	0	0	0
B	4	0	3	4	1	0	0	0
C	2	3	0	3	0	0	0	0
D	1	4	3	0	2	0	0	0
E	0	1	0	2	0	2	0	3
F	0	0	0	0	2	0	5	4
G	0	0	0	0	0	5	0	1
H	0	0	0	0	3	4	1	0

(c) Was berechnet allgemein der Algorithmus von Prim? (1 Punkte)

MST (Minimum SpannBaum)

(d) Folgende Skizze zeigt einen Zwischenschritt zur Erzeugung des Minimalen Spannbaumes (MST) nach dem Algorithmus von Kruskal.



Kanten sortiert:

- AD 1
- BE 1
- HG 1
- AC 2
- DE 2
- EF 2
- BC 3
- CD 3
- EH 3
- AB 4
- BD 4
- FH 4
- FG 5

1. Geben Sie die Tabelle der sortierten Kanten an und markieren Sie, welche Kanten schon verarbeitet worden sind bis zu diesem Zwischenschritt. (3P)

Kosten	Kanten
1	AD
1	BE
1	HG

2. Welche Kanten gehören zum MST? Vervollständigen Sie dazu die Kanten in der Skizze. (2P)

3. Welche Gesamtkosten hat der MST? (1P) 12

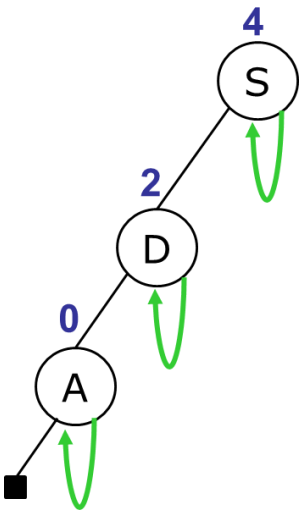
Aufgabe 5 (10 Punkte)
Digitale Suchbäume, Digitaler Trie, Patricia-Baum

Konstruieren Sie für die Schlüsselfolge ADSKL jeweils die Bäume vom Typ Digitaler Suchbaum, Digitaler Trie und Patricia-Tree.

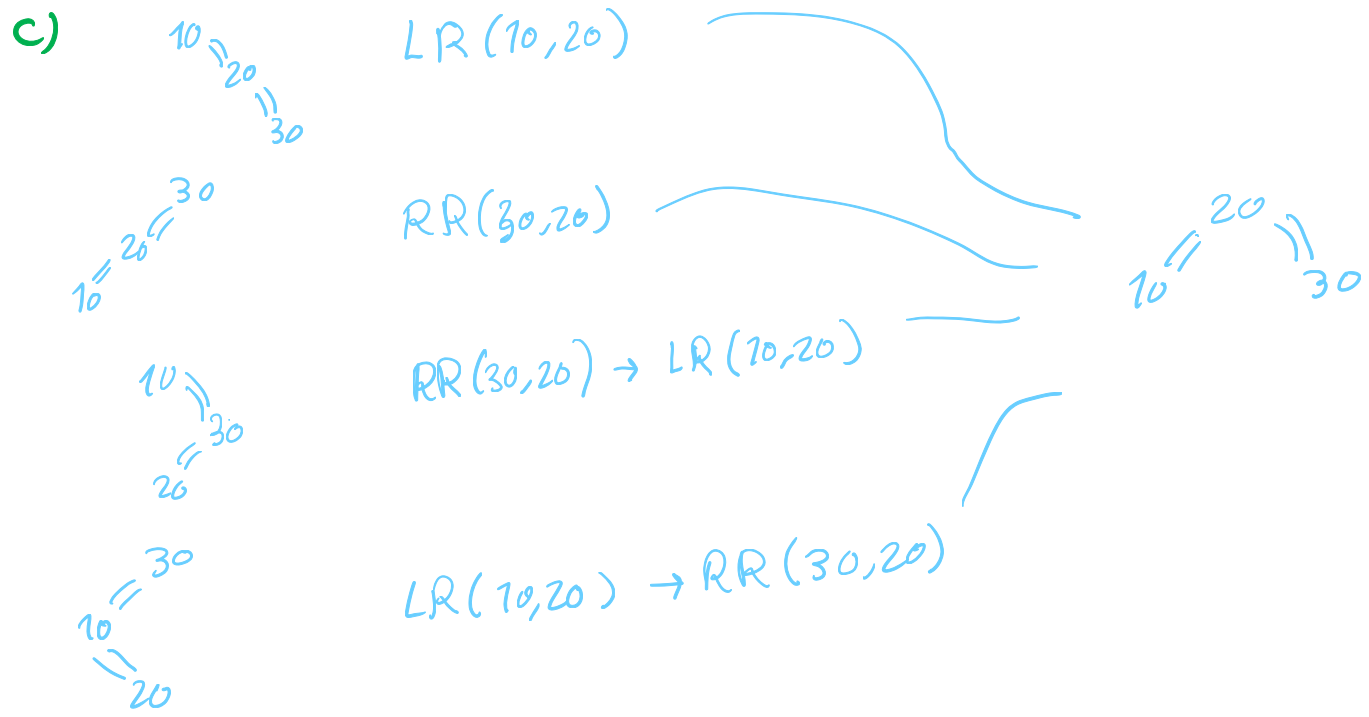
	Dezimal	Binär
A	1	00001
D	4	00100
S	19	10011
K	11	01011
L	12	01100

(a) Digitaler Suchbaum (2 Punkte)
(b) Digitaler Trie (3 Punkte) – skizzieren Sie den Zwischenstand nach jedem Einfügen eines Buchstabens.
(c) Beim Patricia-Baum wurden schon die ersten 3 Schlüssel eingefügt. Fügen sie in diesen bestehenden Baum die nächsten beiden Schlüssel K und L ein. Zeigen Sie jeweils für K und L folgende Schritte:
- Mit welchem Buchstaben wird der einzufügende Schlüssel verglichen?
- Welcher Bitcode entsteht für den neuen Schlüssel. Begründen Sie dies.
- Fügen Sie den Buchstaben in dem Baum ein.

(5 Punkte)



c)


10 = 20 = 30
30 = 20 = 10
10 = 30 = 20
30 = 10 = 20
LR (10,20)
RR (30,20)
RR (30,20) → LR (10,20)
LR (10,20) → RR (30,20)
20 = 10 = 30

STL-Hilfe:

Container, einige wichtige Funktionen und deren Verwendung:

vector<type> Feld

```
vector<int> numbers;
numbers.push_back(3);          // Einfügen am Ende des Feldes
numbers.pop_back();            // entfernt das letzte Element
int last = numbers.back();     // liefert das letzte Element
int first = numbers.front();   // liefert das erste Element
int item = numbers[i];         // liefert das i-te Element
int size = numbers.size();     // liefert die Größe des Feldes
bool empty = numbers.empty(); // true, wenn das Feld leer ist,
                               // andernfalls false
```

list<type> doppelt verkettete Liste (s.Funktionen wie bei vector)

```
list<int> numbers;
numbers.push_back(5); // fügt 5 ans Ende der Liste
numbers.remove(5);    // löscht das Element 5
```

queue<type> Queue

```
queue<int> q; // int – Warteschlange allokieren
q.push(5);    // Element 5 in die Warteschlange
bool leer = q.empty();   // Abfrage, ob Warteschlange leer ist
int element = q.front(); // liefert erstes Element
q.pop() ;     // löscht erstes Element
```

stack<type> Stapel

```
stack<int> nodes; // int - Stapel allokieren
bool ok = nodes.empty();   // überprüfe, ob Stapel leer ist
int element = nodes.top(); // hole Element vom Stapel
nodes.pop();   // entfernt Element vom Stapel
nodes.push(5); // 5 wird auf den Stapel gelegt
int i = (int) nodes.size(); // Anzahl Elemente im Stapel
```

map<keytype, valuetype> (Schlüssel, Referenz)-Container

```
map<int, vector<int> > adjlist;    // Adjazenzliste als Map
int size_adjlist = adjlist.size(); // Anzahl Knoten in der
                                   // Adjazenzliste
vector<int> liste = adjlist[1]; // liefert den Vektor zum
                                // Schlüssel 1
liste[1].push_back(5); // Einfügen von Schlüssel 5 zur Liste
                       // von Schlüssel 1
int size = liste[1].size(); // liefert die Länge der Liste
```