**Formelsammlung:**

* **Write through:**Jeder Schreibzugriff wird auch im Arbeitsspeicher durchgeführt. Jeder Schreibzugriff bedingt einen Zugriff auf den Systembus.
Nachteile: Längere Ausführungszeit
Vorteile: Erheblich einfachere Steuerung
* **Interrupt:** bezeichnet die kurzfristige Unterbrechung der normalen Programmausführung, um eine andere, meist kurze, aber zeitkritische Verarbeitung durchzuführen. Die Interrupt-Service-Routine (ISR) ist ein Stück Maschinenprogrammcode, das die für Unterbrechung notwendigen Aktionen durchführt.
 Vorteile: Laufzeiteffient/ schnelle Reaktion auf Hardware-Events.
 Nachteile: zusätzliche Hardware
* **Polling:** bezeichnet in der Informatik die Methode, den Status eines Geräts aus Hard-oder Software mittels zyklischem Abfragen zu ermitteln.
 Vorteile: benötigt keine zusatzliche Hardware/ Auch Eingänge ohne Interruptfunktion können genutzt werden
 Nachteile: belegt ständig CPU-Leistung
* **Steuerwerk:** Befehlsdekodierung, -adressierung und Ansteuerung der anderen Komponenten
**Rechenwerk:** Führt arithmetisch-logische Operationen, die in Maschinenbefehlen enthalten sind
**Speicherwerk:** Arbeitsspeicher speichert Maschinenbefehle und zugehörige Daten
**Ein-/Ausgabewerk:** Schnittstelle zur Ansteuerung von Peripherie
* **ALU:** Operationswerk, Rechenwerk (Arithmetisch/Logische Einheit).
* **DLX Pipeline: Phasen eines von-Neumann Zyklus**

1) Instruction Fetch (IF): Befehl holen; Program counter (PC) erhöhen.
2) Instruction Decode (ID): Instruktion dekodieren und Werte aus Registern lesen.
3) Execute (EX): Ausführung der Operation/Adressberechnung in der ALU.
4) Memory Access (MEM): Hauptspeicherzugriff; nur für Load-, Store- und Branch-Operationen.
5) Write Back (WB): Ergebnis in Registersatz zurückschreiben.

* **Anzahl Cache-Blöcke** = Cachegrösse in Bytes / Blockgrösse in Bytes
**Anzahl Cache-Sets** = Cachegrösse in Bytes / (Blockgrösse in Bytes\*N)
 (mit N=n-wegassociativ)
* **Offset** = ld(Blockgröße in Bytes) = ld(8) = 3
**Index** = ld(Cachegröße in Bytes/ (Blockgröße in bytes\*N)) = ld(2 hoch16/4) = 14
**Tag** = Adressbreite - Offset - Index = 13
* **Wirklichkeit notwendige Speicherkapazität:** Cachespeicher in Bytes +
**(** **(**Tags [Bit]+1[Valid-Bits]**)** \*Anzahl Cacheblöcke / (8 Bit/Bytes)
* **Die ausschlaggebende Faktoren bei der Auswahl eines Caches :**
1. Der benötigte Speicher
2. Anzahl der benötigten Komparatoren (Logik-Platz)
3. Höhere Flexibilität beim Mapping ( == höhere Cache-Hit-Rate)





* **Programmierparadigmen**:
	+ **Imperativ**: Assembler-Sprachen, FORTRAN, ALGOL
	+ **Prozedural**: Pascal, C
	+ **Objektorientiert**: Smalltalk, C++, Java, C#
* **Unterscheidungsmerkmale von C und C++**:
	+ Syntaktisch ist C++ eine Erweiterung zu C. Dabei stellt die Objektorientierung die grundlegende Erweiterung dar

Nicht jedes C++ Programm lässt sich mit einem C Compiler übersetzen..

* **Speicherverwaltung für lokale Variablen:**
Vorteile : Variablen werden komplett automatisch verwaltet
Nachteil: Platz auf dem Stack für grosse Datenmenge nicht ausreichend
* **Dynamische Speicherverwaltung**:
Vorteile: mehr Speicherplatz als der Stack
Nachteile: der Entwickler muss sich selber kummern um den Speicherplatz zu befreien.





* **Extern:** Variable liegt nicht im zu kompilierenden Quellcode vor, sondern in einer Bibliothek/im Linkerskript und wird beim Linken mit dem Code verknüpft.
* **Static:** Die Variable wird einer festen Speicheradresse zugewiesen, sodass ihr Wert auch verfügbar bleibt, wenn sie vorübergehend nicht mehr sichtbar ist.

🡺 

* **Private**: Diese Elemente sind nur aus Codeteilen heraus zugreifbar, die der Implementierung dieser Klasse dienen, üblicherweise also Methoden-Implementierungen.
* **Public:** Auf diese Elemente ist aus jeder Codepassage heraus Zugriff erlaubt.
* **Protected:** Auf Elemente dieser Zugriffsschutz-Stufe besteht Zugriff von Codeteilen, die entweder die Klasse implementieren, oder für solche, die Klassen implementieren die von der Klasse erben.
* **Friend:** Klassen oder Funktionen, die hinter diesem Schlüsselwort genannt werden, haben vollen Zugriff auf alle Elemente der Klasse.
* **Klasse** erzeugt einen aus mehreren Daten-Membern zusammengesetzten Datentypen sowie einen zugehörigen Satz von Methoden.
* Die Methoden sollen **keinen** **Schreibzugriff auf Klassenattribute besitzen.**
* **iterative** function is one that loops to repeat some part of the code
* **Recursive** function is one that calls itself again to repeat the code.
***Vorteil:*** Vereinfacht die Implementierung von manchen Algorithmen. Kann für bessere Lesbarkeit und Verständnis sorgen.
***Nachteil:*** Schlechtere Performanz/Effizienz, Speicherverbrauch auf dem Stack kann sehr groß werden.
* **Volatile**:In C und C++ spezifiziert dieser Typqualifikator, dass sich der Wert der Variable jederzeit ändern kann.
* **Virtual(wiedernahbarkeit)** als Schlüsselwort für die Methodendeklaration in C++. Bei virtuellen Klassenmethoden wird die Einsprungadresse erst zur Laufzeit ermittelt. Dieses sogenannte dynamische Binden ermöglicht es, Klassen von einer Oberklasse abzuleiten und dabei Funktionen zu überschreiben bzw. zu überlagern.
* **Statische Methoden:** können auch ohne vorherige Instanziierung eines Objektes benutzt werden, indem man sie direkt über den Scope-Operator aufruft: <classname>::<method> Außerdem können statische Methoden nur auf statische Variablen zugreifen und besitzen keinen this-Pointer.
* Nennen Sie vier typische Eigenschaften von objektorientierten Sprachen.
* **Vererbung** Eine Klasse kann von einer anderen Klasse erben. Dabei werden **alle Methoden und Daten**-Member der Eltern-Klasse **an** die **abgeleitete Klasse weitergegeben**.
	+ **Kapselung** Objekt definiert Zugriffsmöglichkeit auf seine Daten und schützt diese (Public, protected, private)
	+ **Polymorphie:** bezeichnet das Prinzip, unterschiedliche Realisierungen einer Methode in abgeleiteten Klassen bereitzustellen
	+ **Abstraktion:** weglassen von Einzelheiten und da Überführen auf etwas allgemeines.
* **Vektor:**


 Vorteil: kontinuierlicher Speicher, indizierter Zugriff.

 Nachteil: Löschen / Einfügen ist aufwendig.

* **Liste:**


**Liste gegebenüber einem Array**

**Vorteile:**
**1.)** Die Größe einer Liste kann zur Laufzeit verändert werden. D.h. die maximale Größe muss zur Compilezeit nicht bekannt sein.
**2.)** Es können effizient Elemente gelöscht und eingefügt werden (ohne umsortieren aller Elemente).

**Nachteile:**
**1.)** Die Zugriffe auf Elemente in einer Liste sind langsamer, da nur ein Zeiger zum Vorgänger und/oder Nachfolger existiert.
**2.)** Höherer Speicherbedarf aufgrund der Zeiger

* **Map:

MAP<> gegenüber einer Vector<> oder List<>**

Map ermöglicht **die Zuordnung** von Namen **zu den zu speichernden Einträgen** und einen effizienten Zugriff über diesen Namen. Die Verwendung von list<> oder vector<> würde außerdem die Definition einer weiteren Datenstruktur erfordern, um Name und Eintrag speichern zu können.

* **Teile und Herrsche:** baut darauf auf, ein Problem immer weiter in kleinere Unterprobleme aufzuteilen bis diese deutlich einfacher bzw im Optimalfall trivial zu lösen sind um sie anschließend wieder zusammenzuführen.
* **Berechnbarkeit:
**
* **& Bitweises UND:** Mit dem UND-Operator werden zwei Operanden bitweise verknüpft. Die Verknüpfung darf nur für Integer-Operanden verwendet werden.
* **&& Logisches UND:** Das Ergebnis des Ausdrucks ist 1, wenn beide Operanden ungleich 0 sind, andernfalls 0. Im Unterschied zum & wird der Ausdruck streng von links nach rechts ausgewertet.
* **Projektmanagement:** ist die Planung, Delegierung, Überwachung und Steuerung aller Aspekte eines Projekts. Dazu gehören die Motivation der Beteiligten, die Projektziele zu erreichen, und zwar innerhalb der zu erwarteten Leistungsziele für Zeit, Kosten, Qualität, Umfang, Nutzen und Risiken.
* **Lastenheft:** Wird vom Auftraggeber erstellt. Enthält die Anforderungen aus Sicht des Auftraggebers/Kunden.
* **Pflichtenheft:** Wird vom Auftragnehmer auf Basis des Lastenhefts erstellt. Darin enthalten ist eine Detaillierung der Kundenanforderungen aus Sicht des Auftragnehmers.
* **Der Projektstrukturplan:** gliedert das Projekt in plan- und kontrollierbare Elemente. Im Rahmen dessen, wird das Projekt in Teilaufgaben und Arbeitspakete unterteilt. Er bildet die Grundlage für die Termin-, Ablauf-, Ressourcen- und Kostenplanung.
* **Der Projektablaufplan:** definiert hauptsächlich den zeitlichen Ablauf eines Projekts - auf Basis der im PSP definierten Arbeitspakete. Normalerweise wird der Projektablaufplan in einem Gantt-Chart/GanttDiagramm dargestellt.
* **Programmierrichtlinien:** (*code conventions)* tragen maßgeblich zur besseren Verständlichkeit und Wartbarkeit von Quellcode bei und können potentielle Fehlerquellen minimieren. Außerdem dienen Sie der Einheitlichkeit des erstellten Quellcodes.
* **Projekt Phasen:**
* **Projektdefinition**: Das Ziel des Projekts wird festgelegt. Kosten, Ausmaß und Zeit werden grob geschätzt.
* **Projektplanung:** In dieser Phase wird das Team organisiert, und es werden Aufgabenpläne, Ablaufpläne, Terminpläne, Kostenpläne festgelegt.
* **Projektdurchführung:** Diese Phase umfasst die Durchführung mit Arbeitspaketen.
* **Projektabschluss:** Die Ergebnisse werden präsentiert und in dokumentierter Form übergeben.

****

* **Zeitliche Parallelität:** Abarbeitung einer Instruktion wird in einzelne Phasen aufgeteilt. Mehrere Befehle werden bzgl. der Phasen zeitlich überlappend abgearbeitet. Beispiel: Pipelining, Superpipelining.
* **Räumliche Parallelität:** Es existieren mehrere Funktionseinheiten, die voneinander unabhängige Befehle parallel bearbeiten können, Beispiel: Superskalarität.
* **CISC (Complex instruction set computing** **) Rechnerarchitekturen:**
*Vorteil:* im Allgemeinen kleinere Programme als RISC, Konstrukte höherer Programmiersprachen teilweise direkt unterstützt.
*Nachteil:* Komplexe, umfangreiche Befehlsätze, komplexe Adressmodi, kein effizientes Pipelining möglich, längere Ausführungszeit.
* **RISC**: Reduced Instruction Set Computers, einfache Befehle, schneller in der Ausführung BSP: SPARC, Power Pc.
* **Addressierungsmodi** einer CPU:
	+ **Register mode**: Der Operand ist in einem der Register gespeichert.
	+ **Direct mode**: Die Speicher-Adresse des Operanden ist in der Instruktion gespeichert.
	+ **Indirect Mode**: Die Speicher-Adresse des Operanden ist in einem Register gespeichert.
	+ **Immediate mode**: Der Operand ist Teil der Instruktion.
	+ **Autoincrement mode**: Die Speicher-Adresse des Operanden ist in einem der Register gespeichert. Der Inhalt dieses Registers wird automatisch inkrementiert nachdem die Adresse gelesen wurde.
* **Statischer Test** ist ein Software-Testverfahren bei dem die Software nicht ausgeführt wird.
* **Dynamischer Test** ist ein Software-Testverfahren bei dem die Software ausgeführt wird. *Fehler die abhängig von dynamischen Laufzeitparametern auftretten, können nicht statisch festgestellt werden.*
* **Black-Box-Tests:** Test von Modulen ohne Wissen über die innere Struktur.
* **White-Box-Tests:** Test mit Wissen über die innere Struktur des Moduls.
* **Verdrängungsstrategie**

**Eine Verdrängungsstrategie ist eine Methode, die entscheidet, welcher Block im Cache ersetzt wird, falls ein neuer Block für neue Daten gebraucht wird.**

*Least Recently Used (LRU):* Der Eintrag, auf den am längsten nicht zugegriffen wurde, wird verdrängt
*Random:* Zufällig ausgewählter Block wird verdrängt
*Least frequently Used (LFU):* Der am seltensten gelesene Eintrag wird verdrängt
*FIFO*: Der jeweils älteste Eintrag wird verdrängt.

* Internen und externen **Architektur eines Prozessors:
Intern**: Definiert die interne Hardware-Architektur des Rechners.
**Extern**: Repräsentiert die Sicht des Programmierers auf den Rechner, den Maschinenbefehlssatz bzw. Instruction Set Architektur (ISA).
* **Register:** Spezielle internen Speicherplätze, die direkt mit der eigentlichen Recheneinheit verbunden sind und die unmittelbaren Operanden und Ergebnisse aller Berechnungen aufnimmt.
* **Komponenten eines Registersatzes:** General Purpose Register, Stackpointer, Programcounter, Instructionregister
* **Operationen, die auf Datenstrukturen** ausgeführt werden können.
**Search(S,k), Insert(S,x), Delete(S,x), Minimum(S), Maximum(S), Successor(S,x), Predecessor(S,x), Size(S)**
* **Call-by-Value** Hierbei wird eine Kopie der übergebenen Struktur erzeugt, die innerhalb der Funktion verwendet und bei Beendigung der Funktion wieder zerstört wird.
***Vorteil***: Die übergebene Struktur wird aufgrund der Kopie nicht verändert. Komplette Trennung der Variablen beim Aufruf
***Nachteil***: Aufgrund der großen Struktur von 1kB erhöht sich der Speicherverbrauch und die Laufzeit aufgrund der Erzeugung der Kopie.
* **Call-by-Reference** Hier wird keine Kopie der Struktur erzeugt, sondern lediglich ein anderer Name für die übergebene Struktur innerhalb der Funktion verwendet.
***Vorteil***: Kein Speicher- und Laufzeit Overhead für die
***Nachteil***: Struktur kann ungewollt überschrieben werden.
* **Call-by-Pointer** Hier wird die Speicheradresse der Struktur übergeben und keine Kopie erzeugt.
Vorteil: Kein Speicher- und Laufzeit Overhead für die Kopie. Zeiger kann verändert werden.
Nachteil: Umständlicher Zugriff über Pointer