

Grundlagen

Dezimalzahlen und Hexadezimal zahlen umwandeln

Binärzahl

1	1	1	1	1	1	0	0	1
265	128	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Hexadezimal

Dezimal-System	Binär-System	Hexadezimal-System
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Beispiel FF50

Stelle	Wert	Potenz (16^x)	Berechnung
F	15	$16^3 = 4096$	$15 \cdot 4096 = 61440$
F	15	$16^2 = 256$	$15 \cdot 256 = 3840$
5	5	$16^1 = 16$	$5 \cdot 16 = 80$
0	0	$16^0 = 1$	$0 \cdot 1 = 0$
		Summe	65360

Daten und Informationen

Unterschied zwischen Daten und Informationen

Begriff	Beschreibung
Daten	Rohwerte, noch ohne Bedeutung - z.B. 43, true, 2025-04-13
Informationen	Bedeutung, die aus den Daten entsteht, wenn sie in Kontext gesetzt werden.

Daten werden zu Information durch:

- Kontext: Woher kommen die Daten? Was bedeuten sie?
- Struktur: Formatierung, Datentypen
- Verarbeitung: Algorithmen, Analysen, Interpretation

Arten von Daten

- Strukturierte Daten: Tabellen, Datenbanken
- Unstrukturierte Daten: Texte, Bilder, Videos
- Halbstrukturierte Daten: XML, JSON

Datenqualität

- Richtigkeit
- Vollständigkeit
- Aktualität
- Konsistenz

ASCII und Unicode

ASCII

- 7 Bit Zeichencodierung (128 Zeichen)
- Standard in frühen Computern, Betriebssystemen und Netzwerken
- Ursprünglich für Englische Texte Entwickelt

Dezimal	Zeichen	Bedeutung
32	(Space)	Leerzeichen
48-57	0-9	Ziffern
65-90	A-Z	Großbuchstaben
97-122	a-z	Kleinbuchstaben
10	\n	Zeilenumbruch
13	\r	Wagenrücklauf

Begrenzungen:

- Kein Support für Umlaute, Sonderzeichen anderer Sprachen, Emojis etc.
- Nur für US-Englisch praktisch verwendbar

Unicode

- Internationaler Standard zur Darstellung aller Schriftzeichen Weltweit
- Ziel: jeder Buchstabe, jedes Emoji, jedes Symbol bekommt eine eindeutige Nummer (Codepoint)
- Beispiel: U+00E4 = "ä"

Unicode hat mehrere verschiedene Kodierungen (Prinzip wie man Unicode-Zeichen als Bytes speichert)

UTF-8	Am verbreitetsten, variable Länge (1-4 Bytes), kompatibel mit ASCII	Effizient, ASCII Kompatibel
UTF-16	2 oder 4 Byte pro Zeichen	Schneller Zugriff, nicht ASCII Kompatibel
UTF-32	4 Byte pro Zeichen, fixe Länge	Selten, eher für Spezialfälle

- Unterstützt alle Sprachen: z.B. Deutsch (ä, ö, ü), Arabisch, Chinesisch
- Emojis, mathematische Symbole, technische Zeichen etc.
- Standard für Web, APIs, Datenbanken und Dateien

Datentypen

- Integer -> Ganzzahlen
- float -> weniger genau (4 Byte)
- double -> höhere Genauigkeit (8 Byte)

- string -> Zeichenkette
- char -> einzelnes Zeichen
- bool -> Wahrheitswert

Softwarequalitätskriterien

Softwarequalitätskriterien sind Merkmale, die beschreiben wie gut eine Software ihre Aufgaben erfüllt - technisch, funktional und nutzerbezogen.

1. Funktionale Qualität (Erfüllt die Software was sie soll)
 - Funktionalität: Sind alle Anforderungen korrekt umgesetzt?
 - Sicherheit: Ist die Software vor unbefugtem Zugriff geschützt?
 - Zuverlässigkeit: Läuft die Software stabil und korrekt?
2. Nicht funktionale Qualität (Wie gut macht die Software das?)
 - Effizienz: Ist sie schnell und Ressourcenschonend
 - Wartbarkeit: Lässt sich der Code leicht ändern und erweitern?
 - Portabilität: Wie kann man sie leicht auf anderen Systemen ausführen?
 - Testbarkeit: Wie einfach ist automatisiertes oder manuelles Testen?
 - Benutzbarkeit (Usability): Ist die Software einfach und angenehm zu bedienen?
 - Wiederverwendbarkeit: Kann man Teile in anderen Projekten wiederverwenden?

ISO/IEC 25010

Kriterium	Beschreibung
Funktionale Eignung	Erfüllt die Software die spezifizierten Aufgaben richtig?
Zuverlässigkeit	Verfügbarkeit, Fehlertoleranz, Wiederherstellbarkeit
Benutzbarkeit	Verständlichkeit, Bedienbarkeit, Benutzerzufriedenheit
Effizienz	Antwortzeiten, Ressourcenverbrauch
Wartbarkeit	Modifizierbarkeit, Analysierbarkeit, Stabilität bei Änderungen
Übertragbarkeit	Anpassbarkeit an neue Umgebungen, Installierbarkeit
Kompatibilität	Interoperabilität mit anderer Software, Co-Existenz

Beispiel für Banking-App

Qualitätskriterium	Bedeutung für die App
Sicherheit	Keine unberechtigten Überweisungen möglich
Zuverlässigkeit	Kein Absturz bei Last
Usability	Nutzer versteht die App sofort
Wartbarkeit	Neue Funktionalitäten (z.B. Apple Pay) leicht einbaubar

Qualitätskriterium	Bedeutung für die App
Effizienz	Schnelle Ladezeiten, geringer Akkuverbrauch

Aufgaben

Für jedes Softwarepaket wird ein Pflichtenheft erstellt. Die darin enthaltenen Anforderungsbeschreibungen müssen den folgenden drei Kriterien genügen:

a.) Vollständigkeit, b.) Eindeutigkeit und c.) Testbarkeit

Erläutern Sie diese drei Begriffe.

1. Eine Anforderung ist vollständig, wenn sie alle notwendigen Informationen enthält, damit Entwickler, Tester und andere Beteiligte genau wissen, was umgesetzt werden soll. Es dürfen keine wichtigen Details fehlen. Beispiel: Wenn eine Funktion beschrieben wird, muss auch angegeben sein, welche Eingaben möglich sind, was genau das Ergebnis ist und was bei Fehlern passieren soll.
2. Eine Anforderung ist eindeutig, wenn sie nur eine einzige Interpretation zulässt. Mehrdeutige oder unklare Formulierungen wie "schnell", "benutzerfreundlich" oder "in der Regel" sind zu vermeiden. Beispiel: Statt "Das System soll schnell starten" sollte stehen: "Das System soll innerhalb von 5 Sekunden nach dem Einschalten betriebsbereit sein"
3. Eine Anforderung ist testbar, wenn man objektiv überprüfen kann, ob sie erfüllt wurde. Dafür müssen klare messbare Kriterien vorliegen. Beispiel: Eine testbare Anforderung könnte lauten: "Das System soll 1000 Anfragen pro Minute ohne Fehlermeldung verarbeiten können".

Welche Konsequenzen drohen bei Verletzung dieser Kriterien?

Fehlerhafte Anforderungen führen fast immer zu höheren Kosten, längerer Projektlaufzeit, schlechter Qualität und unzufriedene Kunden.

Beurteilen Sie folgende Anforderung im Licht der drei Kriterien:

Die Länge des Güterzugs beträgt bei Einsatz einer einzelnen Lokomotive maximal 30 Waggon. Aus ökonomischen Gründen darf er aber auch nicht zu kurz sein.

a => Es fehlt ganz klar an Informationen, was ist "zu kurz"

b => Es ist nicht eindeutig, ob sich diese Anforderungen auf die Anzahl der Waggon oder die Länge des Zuges selbst beziehen

c => Durch fehlenden Kontext und fehlende Information ist diese Anforderung nur bedingt testbar.

Die zu erstellenden Softwarelösungen sollen effektiv und effizient sein

Grenzen Sie diese Begriffe voneinander ab.

Effektivität beschreibt Ob das Ziel/die Anforderung überhaupt erreicht wird.

Effizienz beschreibt wie gut in dem Falle die Software die Anforderungen ausführt. Effizienz beschreibt dagegen wie

Nennen Sie zwei exemplarische Möglichkeiten, ein Programm effizienter zu gestalten

- Optimierung von Algorithmen => Optimierung der Rechenzeit bei großen Datenmengen
- Reduzierung des Speicherverbrauchs => Optimierung der Datenstruktur je nach Größe der Datenmengen

Bei der Verwaltung des Zugaufkommens fallen große Datenmengen an. Ihr Team diskutiert darüber, ob zur Sortierung dieser Daten ein quadratisches oder ein logarithmisches Verfahren vorzuziehen sei. Nennen Sie jeweils einen Vorteil und einen Nachteil quadratischer und logarithmischer Sortierverfahren.

Quadratische Sortierverfahren (z.B. Bubble Sort, Selection Sort):

Vorteil:

- Einfachheit und Implementierung: Quadratische Algorithmen sind einfach zu verstehen und schnell zu Implementieren, da sie keine komplexen Datenstrukturen oder rekursive Aufrufe erfordern. Dies macht sie besonders für kleinere oder weniger komplexe Anwendungen nützlich.

Nachteil

- Schlechte Skalierbarkeit: Quadratische Algorithmen haben eine Zeitkomplexität von $O(n^2)$, was bedeutet, dass die Laufzeit mit zunehmender Datenmenge stark ansteigt. Bei großen Datensätzen wird die Ausführung extrem langsam, was sie bei großen Datenmengen ineffizient macht.

Logarithmische Sortierverfahren (z.B. MergeSort, QuickSort)

Vorteil:

- Hervorragende Skalierbarkeit: Logarithmische Sortierverfahren wie MergeSort und QuickSort haben eine Zeitkomplexität von $O(n \log n)$. Das bedeutet, dass die Laufzeit mit wachsender Datenmenge viel langsamer zunimmt als bei quadratischen Algorithmen. Sie sind daher sehr gut geeignet für Verarbeitung großer Datenmengen.

Nachteil

- Komplexität der Implementierung: Diese Algorithmen sind komplexer zu implementieren und erfordern oft zusätzliche Speicherressourcen (z.B. bei MergeSort). Außerdem können sie in spezifischen Fällen (z.B. bei schlecht verteilten Daten) durch ungünstige Rekursionstiefen langsamer werden.

Um die Codeerstellung im Team zu vereinheitlichen, gelten in Ihrem Unternehmen Codierungsrichtlinien.

Für die Schreibweise von Variablennamen gilt die Camel-Case-Notation. Erläutern Sie diesen Begriff anhand eines Beispiels.

CamelCase beschreibt das Verwenden von Großbuchstaben für jedes Wort innerhalb einer Variable.

Nennen Sie vier weitere Beispiele für solche Gestaltungsrichtlinien

- Einheitliche Kommentierung und Dokumentation wie z.B. Methoden oder Inline-Kommentare
- Verwendung von Konstanten statt "Magic Numbers". Anstatt feste Werte direkt im Code zu verwenden, sollen diese in benannten Konstanten gespeichert werden.
- Fehlerbehandlung und Logging. Fehler müssen immer korrekt abgefangen und behandelt werden, und es müssen aussagekräftige Logs erstellt werden, um die Ursachen von Problemen schnell nachvollziehen zu können.
- Vermeidung von Code-Duplizierung, der Code muss so strukturiert sein, dass redundante oder doppelte Logik vermieden wird

Revision #6

Created 13 April 2025 10:14:43 by Admin

Updated 27 April 2025 22:10:16 by Admin