Datentypen

Ganzzahlvariablen

- Char:
  - speichert ganze Zahlen zwischen $-128 = -2^7$ und $127 = 2^7-1$
  - werden in der Regel als Buchstabe interpretiert

```
char c = 65, buchstabe = 'A';
```

- Integer:
  - speichert ganze Zahlen zwischen -2^{31} und $+2^{31}-1$

```
int i, ganzeZahl;
```

- Sonstige (nicht häufig benutzt)
  - long, long long, short etc verhalten sich wie integer-Variablen, sie haben lediglich einen anderen Wertebereich

Datentypen

Ganzzahlvariablen

- signed / unsigned
  - wenn nichts gesagt wird, sind chars und integer immer signed (vorzeichenbehaftet), man kann jedoch chars und integer ohne vorzeichen deklarieren, um den positiven Wertebereich zu verdoppeln

```
unsigned char positiverChar;
```

```
unsigned int positiverInt;
```

Wertebereich	signed	unsigned
char	$-2^7 \ldots +2^7-1$	$0 \ldots +2^8-1$
int	$-2^{31} \ldots +2^{31}-1$	$0 \ldots +2^{32}-1$

Datentypen

Kommazahlvariablen

- Float / Double:
  - speichert Kommazahlen, wobei double doppelt so genau ist wie float (genauer Zahlenbereich unwichtig)
  - kann viele Zahlen nicht 100% genau speichern, z.B die Zahl 0.6

    (→ 2tes Semester)

```
float f, kommaZahl; double strecke;
```

Datentypen

Zeiger

- Pointer sind Variablen, die auf andere Variablen zeigen, indem sie die Speicheradresse der Variable speichern, auf die sie gerade
- In C benutzt man den *-Operator als „Zeigen-auf-Operator“, dem gegenübergestellt ist der &-Operator, der als „Adresse-von-Operator“ interpretiert werden kann

Adresse	&a = 992	&b = 996	&c = 1000	&d = 1004
Wert	a = *c = 4	b = *d = 2	c = 992	d = 996

- & und * heben sich gegenseitig auf

&(*pt) = *(&pt) = pt

Datenstrukturen

Arrays

- Arrays dienen der Organisierung mehrerer Variablen gleichen Typs
- z.B. Speicherung 10 ganzer Zahlen oder 15 Buchstaben

a[0] = 3	a[1] = 1	a[2] = 4	a[3] = 1	a[4] = 5	a[5] = 9

ein Array aus Integern

Deklaration: **int a[6];**

Stephan Alt
stephan.alt@rwth-aachen.de

Datenstrukturen

Arrays

- Arrays können aus jedem Datentyp geformt werden
- auch das Array ist ein Datentyp, sodass Arrays aus Arrays gebildet werden können

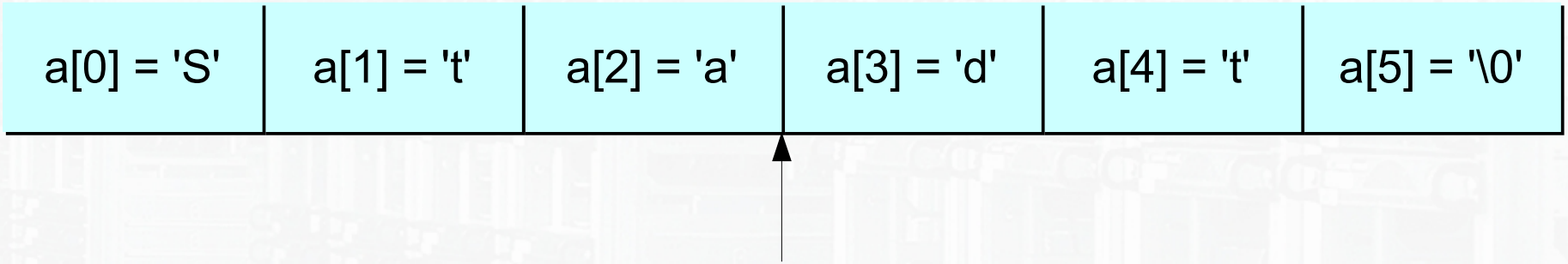
a[0][0] = 3	a[0][1] = 1	a[0][2] = 4	a[0][3] = 1	a[0][4] = 5	a[0][5] = 9
a[1][0] = 2	a[1][1] = 6	a[1][2] = 5	a[1][3] = 3	a[1][4] = 5	a[1][5] = 8
a[2][0] = 9	a[2][1] = 7	a[2][2] = 9	a[2][3] = 3	a[2][4] = 2	a[2][5] = 3

- Dies ist ein Array, welches 3 Arrays der Länge 6 enthält, welches seinerseits Integer-Variablen speichert

Datenstrukturen

Arrays

- In C werden Wörter (Strings) in Arrays aus chars gespeichert
- wichtig dabei ist, dass immer ein Zeichen mehr benötigt wird als der String Zeichen enthält



der string „Stadt“

- Dabei ist das Zeichen '\0' = 0 definiert als „Ende der Zeichenkette“

Datenstrukturen

Arrays

- Arrays sind in C als Pointer implementiert

arr[i] entspricht *(arr+i)	&arr[i] entspricht arr+i

- Dieses Wissen ist aber nicht unbedingt nötig um vernünftig mit Arrays zu arbeiten

Adresse	**888**	**892**	**896**	**900**
Wert	**a[0] = 4**	**a[1] = 2**	**a[2] = 1**	**a[3] = 0**

a = 888	*(a+1) = 2	&a[2] = 896	a[3] = 0

Datenstrukturen

Structs

- Structs speichern wie Arrays mehrere Variablen gleichzeitig
- jedoch können mit Structs verschiedene Datentypen gespeichert werden

```
struct person{
    char vorname[20];
    char nachname[20];
    int telefonnummer;
    double groesse;
    };
struct person hans;
```

hans	hans.vorname	„Hans“
	hans.nachname	„Mustermann“
	hans.telefon	0241 123456
	hans.groesse	1.93

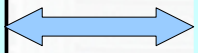
Datentypen

typedef

- wird ein bestimmter Datentyp in einem Programm regelmäßig benutzt, lohnt es sich diesem Typen aus Faulheit einen eigenen Namen zu geben

```
typedef unsigned char uchar;
```

```
unsigned char a;
```



```
uchar a;
```

```
typedef struct {
    double x;
    double y;
} koordinaten;

koordinaten punkt1, punkt2;
```