References

- When a variable is declared as a reference, it becomes an alternative name for an existing variable. A variable can be declared as a reference by putting '&' in the declaration (geeksforgeeks).
- Eine Referenz ist ein Verweis auf ein Objekt. Eine Referenz ist damit ein Aliasname für ein bereits bestehendes Objekt (Wikipedia).
- References allow us to build an alias for an already existing object (Vorlesung).

References: Example 1

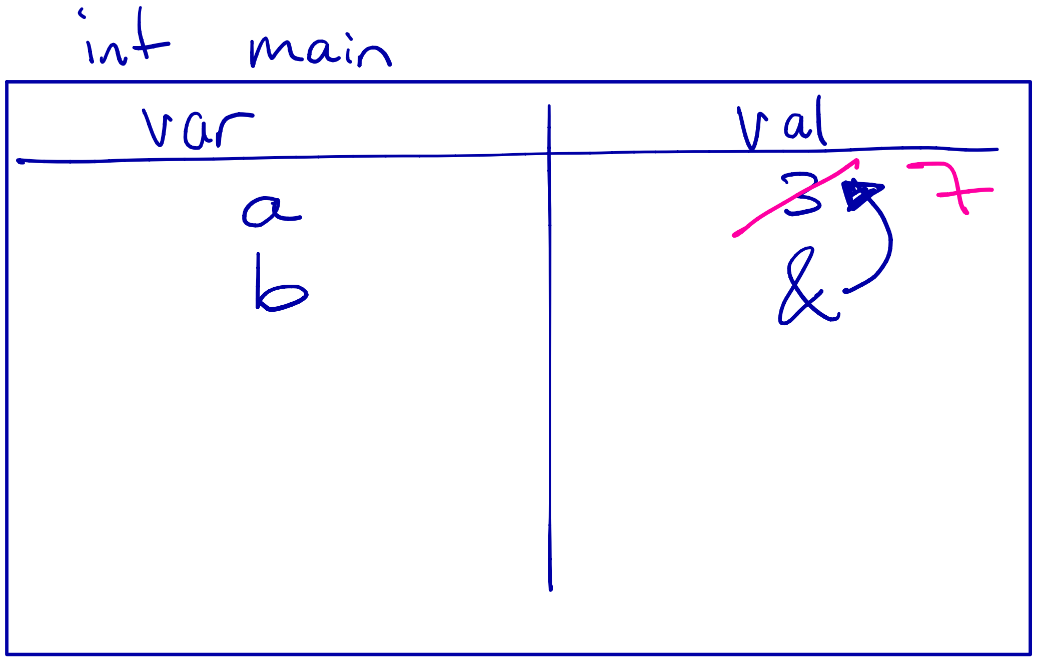
```cpp
int a = 3;
int& b = a;
b = 7;
std::cout << a; // Output = ?
```

References: Example 1

```
int a = 3;
int& b = a;
b = 7;
std::cout << a; // Output = 7
```

References: Example 1

```
int a = 3;
int& b = a;
b = 7;   *
std::cout << a; // Output = 7
```



References: Example 2

```cpp
void foo (int i) {
    i = 5;
}
int main () {
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

pass by value

```
int foo(int i){
    i=5;
    return i;
}
```

```
i = foo(i);
```

int main

var	val
i	4

void foo

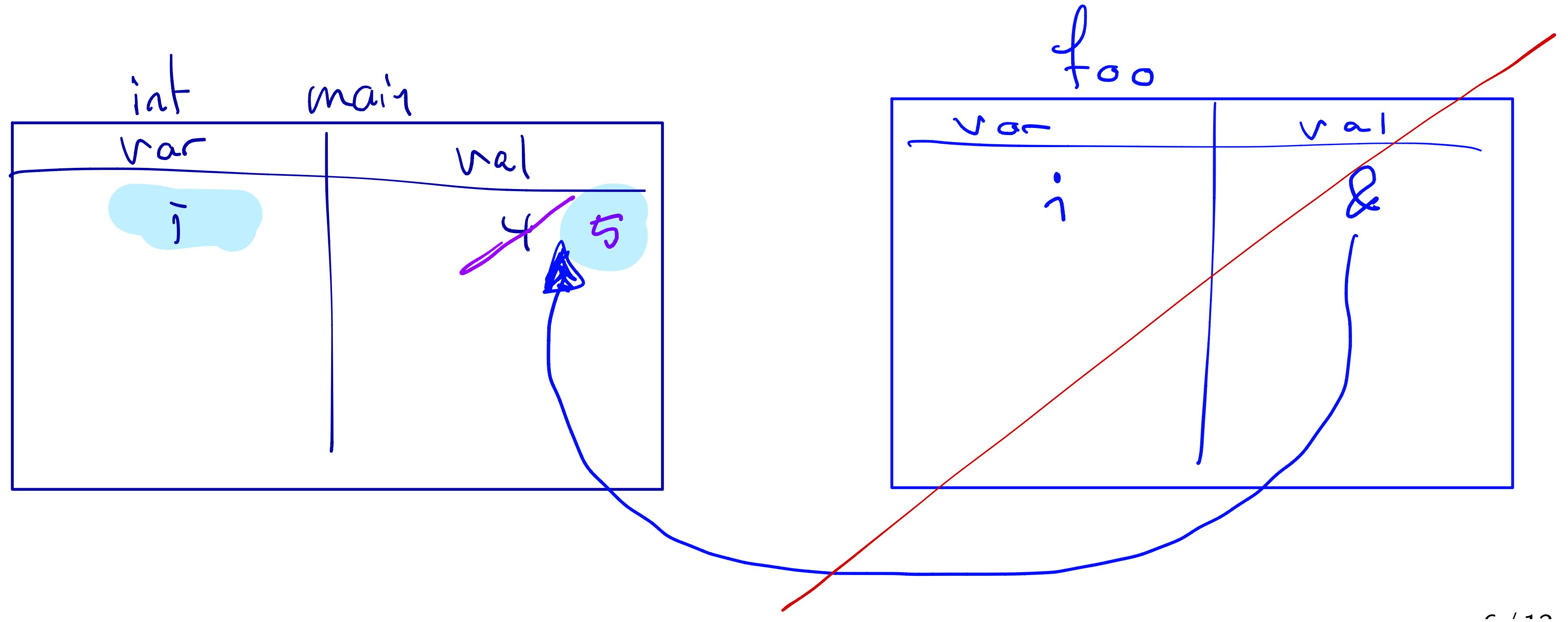
var	val
i	~~4~~ 5

References: Example 3

pass by reference

```
void foo (int& i) {
    i = 5; *
} *
int main () {
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

```
foo2( int& a){
  a=5;
}
```



Why do we need references?

→ mehrere return werte

```
int solve_quadratic_equation(
    const double a, const double b, const double c,
    double& s1, double& s2){...}
```

→ (sehr) grosse objekte

```
void read_i (Vector& v, unsigned int i);
```

```
// Error: copying std::cout is impossible
std::ostream o = std::cout;
// this now works
std::ostream& o = std::cout;
```

References as return types

nur return ref wenn auch schon als ref passed

int f(- - -)

return a ;

```
int& increment (int& m) {
  return ++m;
}
int main () {
    int n = 3;
    increment (increment (n));
    std::cout << n << "\n"; // outputs 5
    return 0;
}
```

inc(&)

Exercise 1

Exercise 1

$a += b \iff a = a + b$

m

(a)

What is the output of the program for the following variant of `foo`?

1 2

```
int foo (int& a, int b) {
  a += b;          // a = 1      / 2
  return a;        // return 1
}
```

```
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<5; ++i) {
    b = foo (a, b);      // b = 1 ,   a = 1
    std::cout << b << " ";
  }
  return 0;
}
```

nach i = 1:
a = 2
b = 2

Exercise 1

(a)

What is the output of the program for the following variant of `foo`?

```
int foo (int& a, int b) {
  a += b;
  return a;
}
```

```
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<5; ++i) {
    b = foo (a, b);
    std::cout << b << " ";
  }
  return 0;
}
```

```
1 2 4 8 16
```

Exercise 1

(b)

What is the output of the program for the following variant of `foo`?

```
int foo (int a, int b) {
  a += b;
  return a;
}
```

```
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<5; ++i) {
    b = foo (a, b);
    std::cout << b << " ";
  }
  return 0;
}
```

Exercise 1

(b)

What is the output of the program for the following variant of `foo`?

```
1 1 1 1 1
```

```
int foo (int a, int b) {
  a += b;
  return a;
}
```

```
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<5; ++i) {
    b = foo (a, b);
    std::cout << b << " ";
  }
  return 0;
}
```

Exercise 1

(c)

What is the output of the program for the following variant of `foo`?

```
int foo (int a, int& b) {
  a += b;
  return a;
}
```

```
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<5; ++i) {
    b = foo (a, b);
    std::cout << b << " ";
  }
  return 0;
}
```

Exercise 1

(c)

What is the output of the program for the following variant of `foo`?

```
1 1 1 1 1
```

```
int foo (int a, int& b) {
  a += b;
  return a;
}
```

```
int main() {
  int a = 0;
  int b = 1;
  for (int i=0; i<5; ++i) {
    b = foo (a, b);
    std::cout << b << " ";
  }
  return 0;
}
```

Characters: char

0 - 127

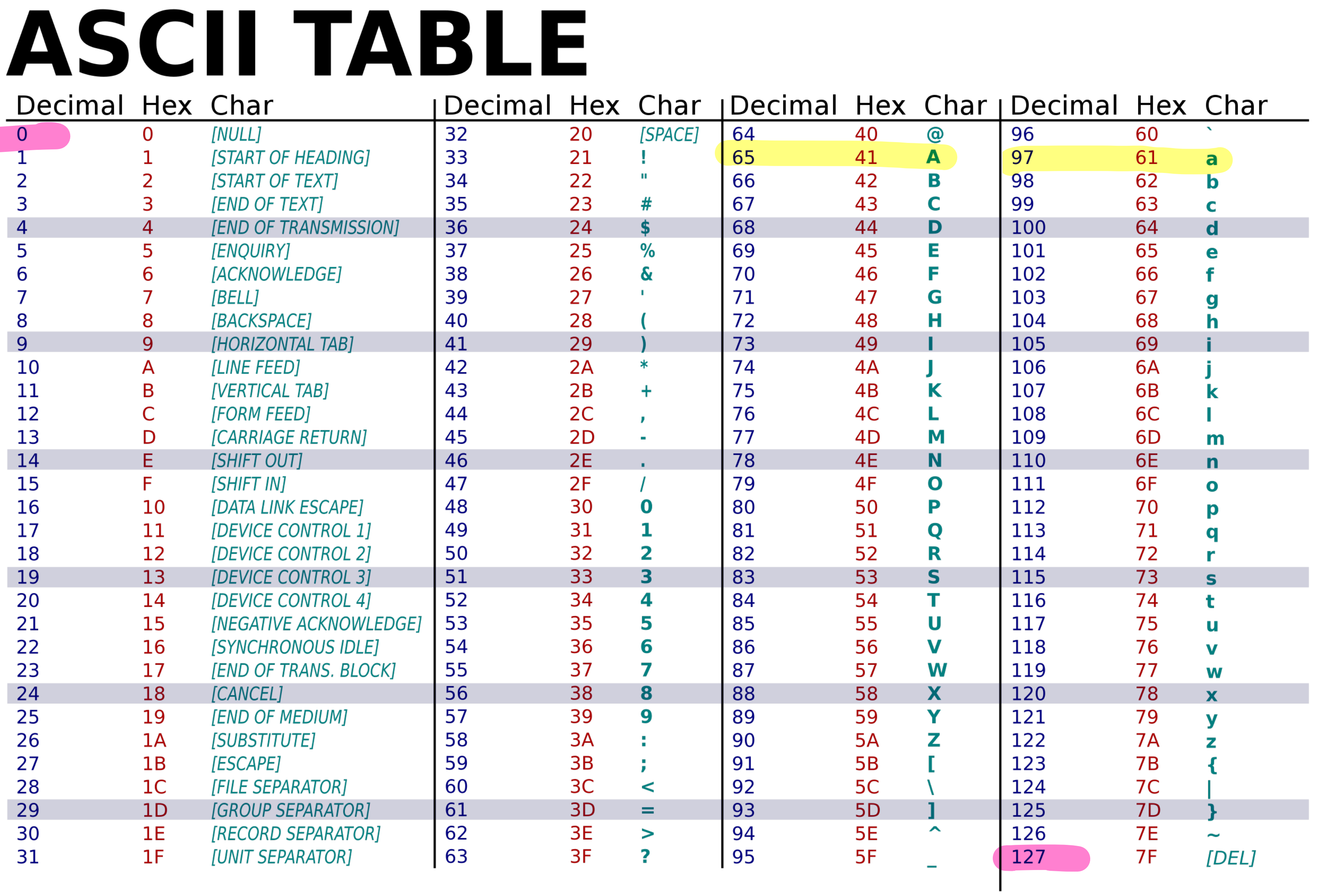
In C and C++, an integer (ASCII value) is stored in char variables rather than the character itself. For example, if we assign 'h' to a char variable, 104 is stored in the variable rather than the character itself. It's because the ASCII value of 'h' is 104.

`https://www.programiz.com/cpp-programming/char-type`

Characters: ASCII table

cout << A − a

cout << 65 − 97

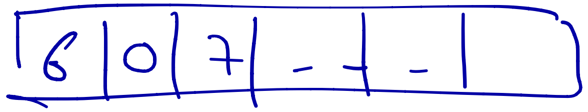


ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Wikipedia

std::vector



```
#include <vector>
...
std::vector<type> vec_name;

// some examples
std::vector<bool> first; // empty vector of bools
std::vector<int> second (4,0); // 4 ints with value 0
std::vector<int> third (second); // a copy of second
```

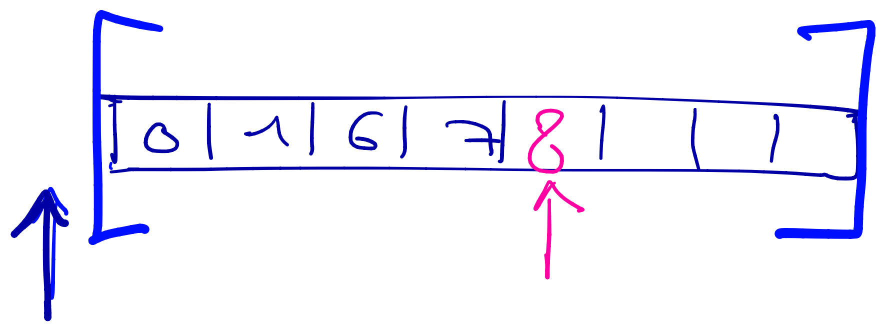
`https://cplusplus.com/reference/vector/vector/vector/` and

`https://en.cppreference.com/w/cpp/container/vector/vector`

Nützliche Funktionen von std::vector

Auf `https://www.geeksforgeeks.org/vector-in-cpp-stl/` werden unter "Capacity", "Modifiers" und "Element access" nützliche Funktionen beschrieben die ein Vektor euch anbietet. Hier einige davon:

- `push_back(element)`
- `at(index)`
- `size()`



```
for (unsigned int i = 0; i < letters.size(); ++i) {
    std::cout << letters.at(i);
}
```

```
const int a = 5;


void print( int& a ) { ... }
          (const int & a)
```

Course and Exercise Feedback

captionEnter Caption