C Programming
1DT301

Lecture 1

[OPNOVA]
ENGINEERED INNOVATION

Intro

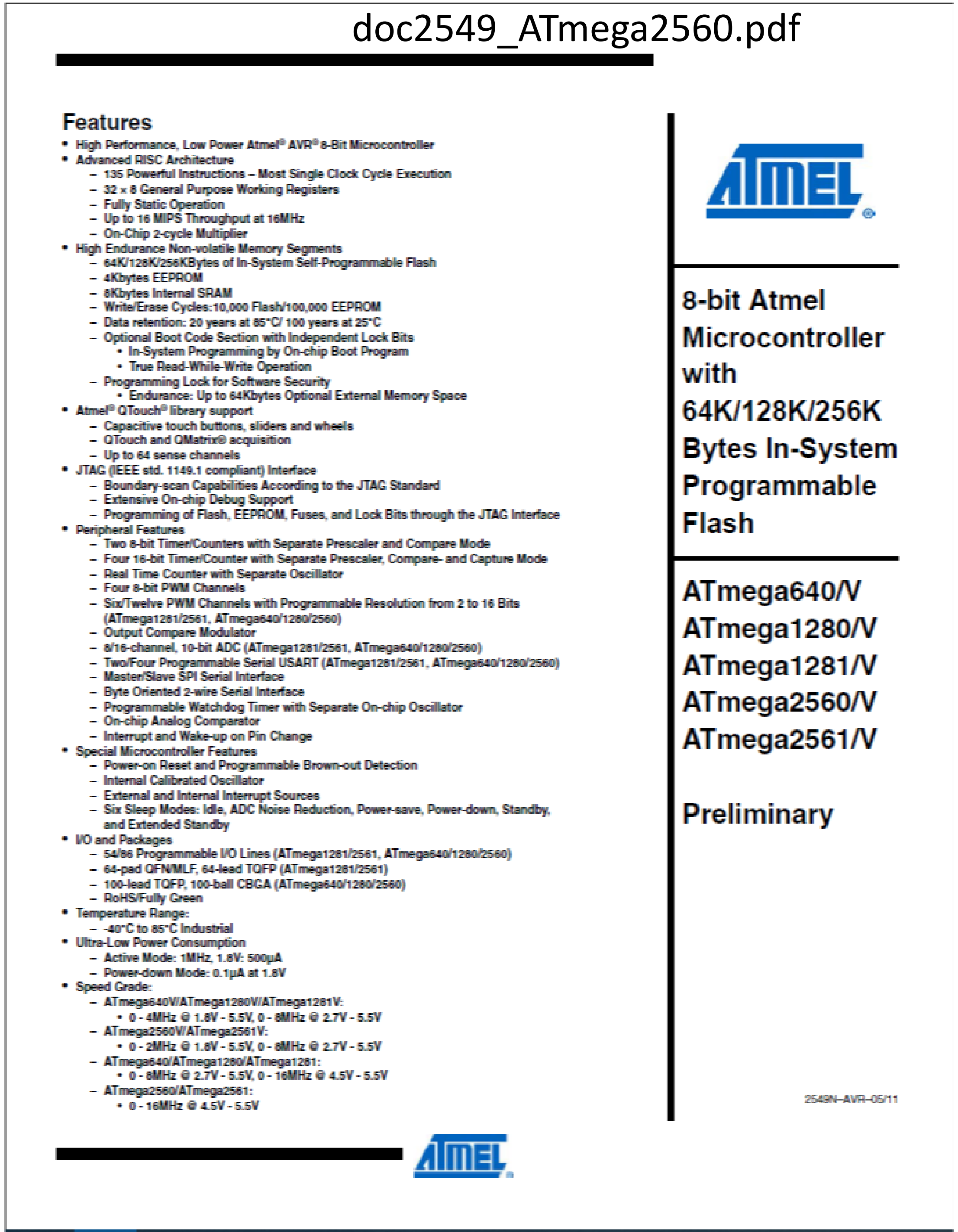
Purpose

- C Programming language
- Preparations for Lab 6
- 8 h

Do You know this one?

Have a look at chapter 22!

doc2549_ATmega2560.pdf



Features

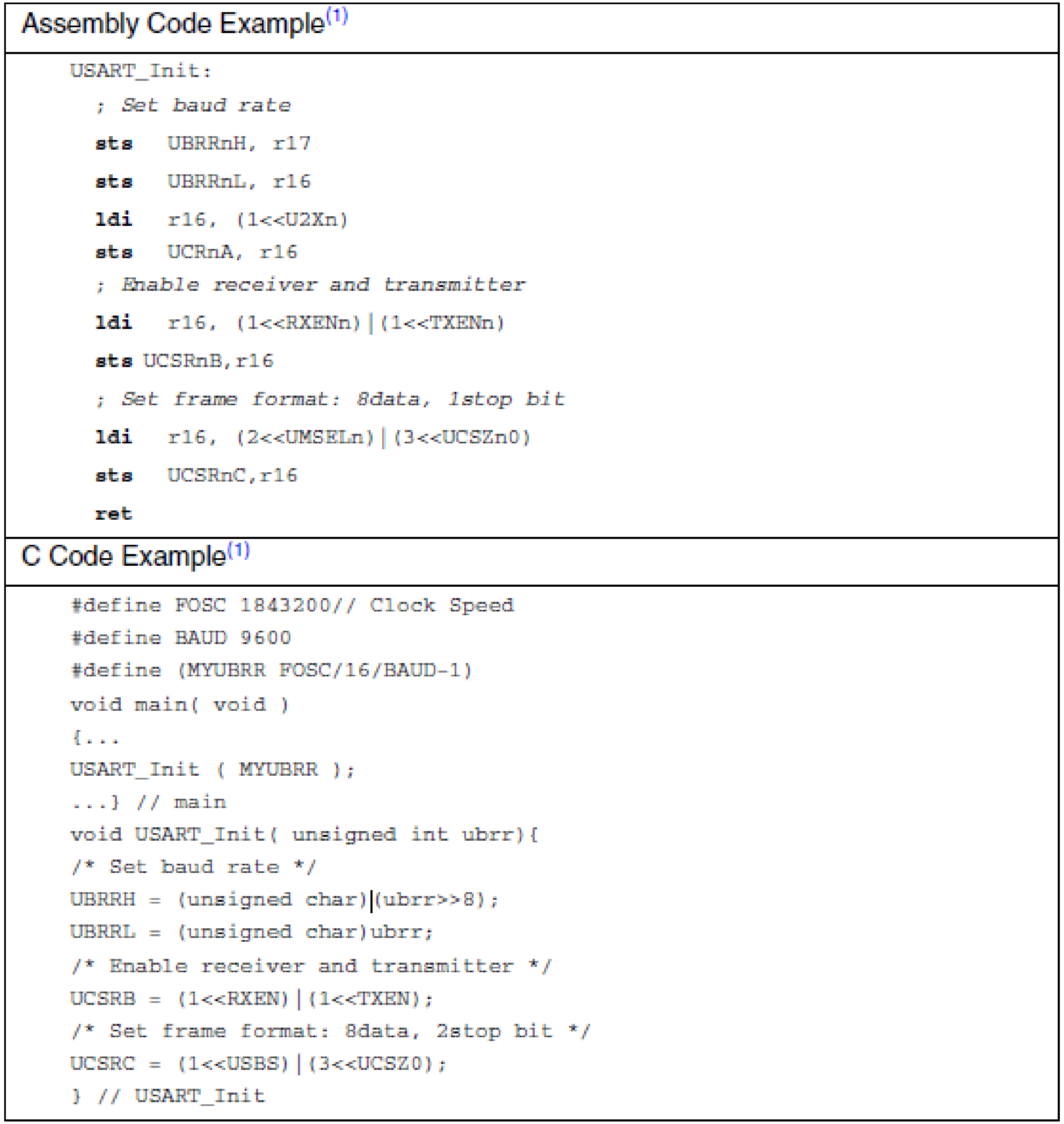
- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

Preliminary

2549N–AVR–05/11



Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  sts   UBRRnH, r17
  sts   UBRRnL, r16
  ldi   r16, (1<<U2Xn)
  sts   UCRnA, r16
  ; Enable receiver and transmitter
  ldi   r16, (1<<RXENn)|(1<<TXENn)
  sts UCSRnB,r16
  ; Set frame format: 8data, 1stop bit
  ldi   r16, (2<<UMSELn)|(3<<UCSZn0)
  sts   UCSRnC,r16
  ret
```

C Code Example[1]

```
#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define (MYUBRR FOSC/16/BAUD-1)
void main( void )
{...
USART_Init ( MYUBRR );
...} // main
void USART_Init( unsigned int ubrr){
/* Set baud rate */
UBRRH = (unsigned char)(ubrr>>8);
UBRRL = (unsigned char)ubrr;
/* Enable receiver and transmitter */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Set frame format: 8data, 2stop bit */
UCSRC = (1<<USBS)|(3<<UCSZ0);
} // USART_Init
```

Lab 6 (excerpt)

Goal for this lab:
The aim is to use STK600 for communication with an external device. In this case, a display that has its own protocol and the documentation is not entirely clear. Description of the protocol (CyberTech_Display.pdf) and installation manual are available on the website (mymoodle.lnu.se); see also Lecture 4 on Embedded C Programming.

The program should be written in C.

Task 1: Write a program that writes a character on the CyberTech Display.
Any character can be displayed. The display is connected to the serial port (RS232) on the STK600. Communication speed is 2400 bps.

Task 2: Write a program that writes characters on all text-lines on the CyberTech Display.
The program will write to all three rows.

Task 3: Write a program that change text strings on the display.

Lab 6

Task 4: Write a program that communicates with both the terminal and the display.
Since we only have one serial port, we must make a special cable, so that the STK600 receive unit is connected to the terminal (PuTTY, for instance) and transmit is connected to the display. Text can be entered at the terminal. End of line with a special character that you choose. It should also be possible to enter address on the screen to display text.

Task 5: Write a program for text input.
When text is inserted it should be possible to choose the address (1 - 9) of the display and send it via the serial port.

Try to write/draw an overview for this.

(We will get back to this at the end of the 4th lecture)

Programming in C

Content

- C (Minimal basics)
- Useful constructs when programming in small embedded systems
- No Analysis & Design in this course!

Litterature

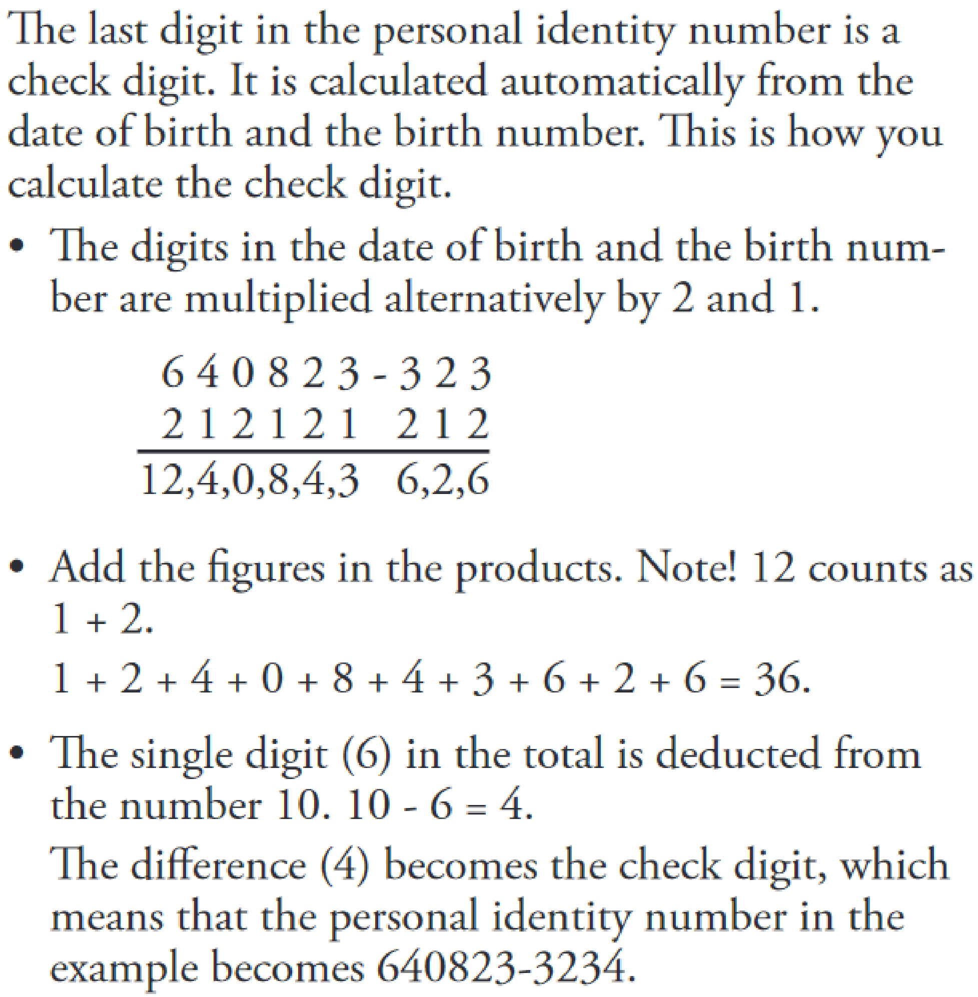
- The C Book, http://publications.gbdirect.co.uk/c_book/thecbook.pdf
- Learning GNU C, http://nongnu.askapache.com/c-prog-book/learning_gnu_c.pdf
- The GNU C Reference Manual, http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf

Programming-In-General

- Think first → Program later
  - In general don’t use Trial-by-error (or Learning-by-doing)
- Mental image of things & Algorithmic thinking
  - Building a house where others would live...
- Abstraction → Concrete

(Have a look at the concept of Computational Thinking)

Example of an algorithm



The last digit in the personal identity number is a check digit. It is calculated automatically from the date of birth and the birth number. This is how you calculate the check digit.

- The digits in the date of birth and the birth number are multiplied alternatively by 2 and 1.

```
  6 4 0 8 2 3 - 3 2 3
  2 1 2 1 2 1   2 1 2
 12,4,0,8,4,3   6,2,6
```

- Add the figures in the products. Note! 12 counts as 1 + 2.

  1 + 2 + 4 + 0 + 8 + 4 + 3 + 6 + 2 + 6 = 36.

- The single digit (6) in the total is deducted from the number 10. 10 - 6 = 4.

  The difference (4) becomes the check digit, which means that the personal identity number in the example becomes 640823-3234.

There is no programming language here!

http://www.skatteverket.se/privat/folkbokforing/omfolkbokforing/personnumretsuppbyggnad.4.18e1b10334ebe8bc80001502.html

C

- Small & terse language (few keywords and constructs)
- Low-level (mostly)
- “Subset” of C++
- UNIX
- Dennis Ritchie[1]
- ALGOL60 → BCPL→ B → C[2]

[1] cm.bell-labs.com/cm/cs/who/dmr (search for `The Development of the C Language')
[2] http://www.levenez.com/lang/

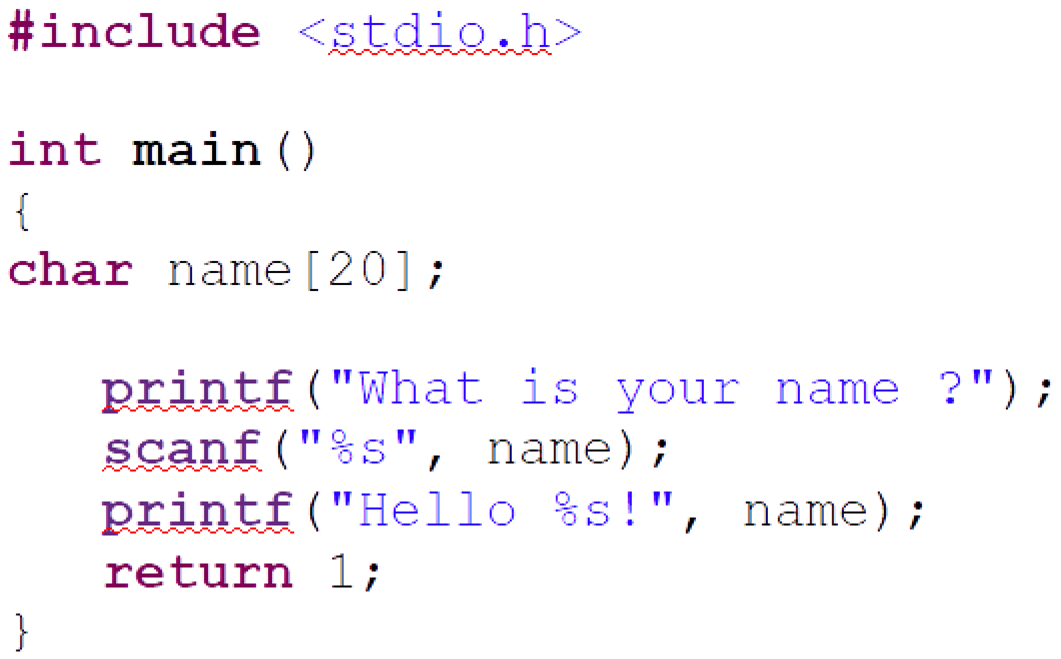
Typical Uses

- Real-time systems
- Embedded systems
- Server applications
- Desktop applications (console)

Keywords

```
auto
break
case
char
const
continue
default
do
double
else
enum
extern
float
for
goto
if
inline
int
long
register
restrict
return
short
signed
sizeof
static
struct
switch
typedef
union
unsigned
void
volatile
while
_Bool
_Complex
_Imaginary
```

A C Program

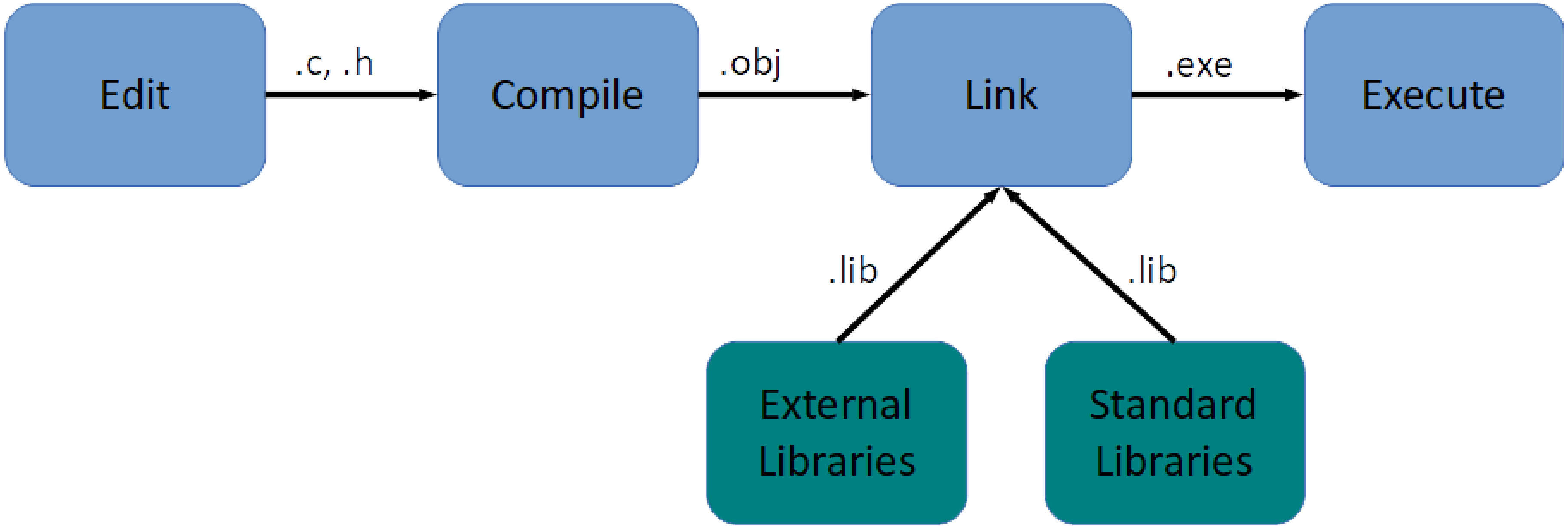


```
#include <stdio.h>

int main()
{
char name[20];

    printf("What is your name ?");
    scanf("%s", name);
    printf("Hello %s!", name);
    return 1;
}
```

Workflow



There is more to it than this!

Declaration vs. Definition

- Declaration
  - A place where a symbol (name+type) is stated.
  - Compiler can do its job with declarations only.
- Definition
  - The place where the symbol is created or assigned storage.
  - Linker needs definitions.
  - Missing definitions will produce (weird) linker errors!
- An identifier can be both declared and defined at the same time.

Headers

- Contains declarations
  - So there need to be definitions somewhere!
- AKA *Include files* (and similar Assembly language `.include` directive).
- Standard libraries (need .lib-files!)
- Modularisation[1]
- Abstract Data Types[1]

[1] Not in this course.

Headers

- File (.h)
- Referred to by using #include
- #include <stdio.h> ↔ #include ”stdio.h”
- Custom made header files.
    - Typically #include ”filename.h”
- Problem with circular includes.

Some standard header files

Include	Content
assert.h	Diagnostics.
math.h	Mathematical functions and macros
stdio.h	Input and output functions and macros.
stdlib.h	Number conversions, storage allocations etc…
string.h	String handling.
time.h	Manipulating time and date.

Call-By-Value

- When calling a function, You receive a copy of the caller's value(s).
- Simulates Call-By-Reference by using pointers.

Preprocessor

- Performs
  - Inclusion of files (#include)
  - Macro substitution. This means textual replacement! Watch out!
  - Conditional compilation
- Starts with the # character

(More on this later)

Variables & Types

Variable

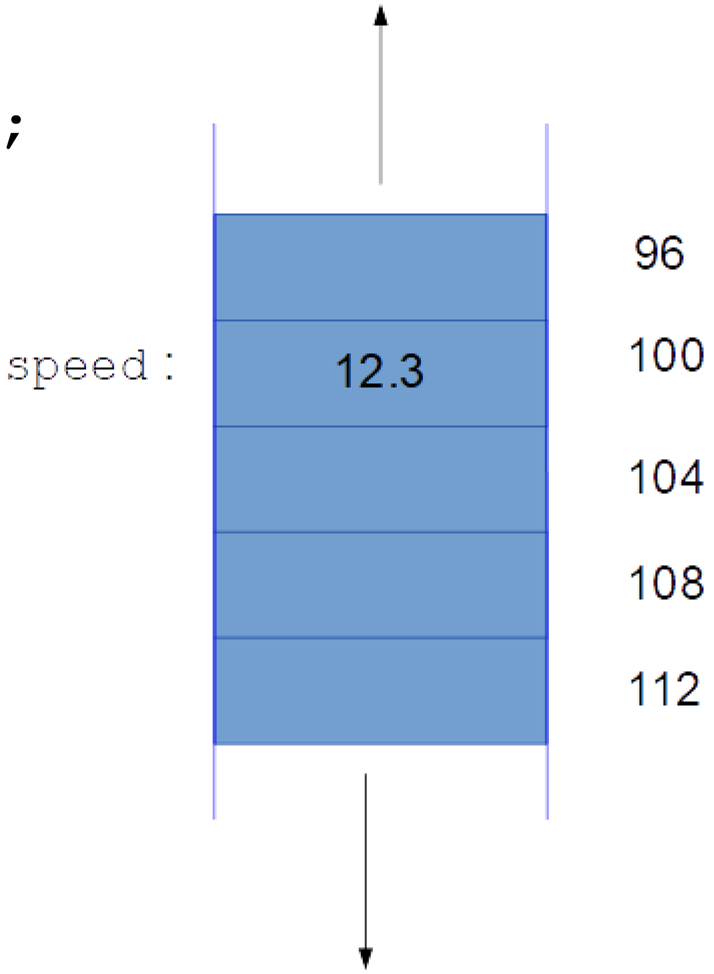
- *A reserved place in memory*
- Identifier (name)
- Type (size !)

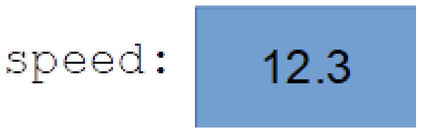
Type
Identifier

```
float speed;
speed = 12.3;
```

Variables

```
float speed;
speed = 12.3;
```





Think in terms of memory locations and spaces!

Scope

- Life of a variable (identifiers).
- Storage classes
  - auto, register, extern, static

Constants

- Not a variable :)
- Prevent from overwriting (hopefully...)
- const int MYCONST = 23;
- #define MYCONST 23

Enumeration Constants

- A set of integers represented by identifiers.
  - Instead of a separate const declarations.
- User-defined type.

```
enum card {CLUBS, HEARTS, SPADES, DIAMONDS};

enum card {CLUBS = 0, HEARTS, SPADES, DIAMONDS};

enum card {CLUBS, HEARTS=2, SPADES = 7, DIAMONDS};
```

Types

- Implementation dependent
  - `int` on one vendor is implemented as 16-bits, another vendor uses 8-bits.
- Misuse of → Many times the result is a *Undefined behavior.*
- `void` - means 'an object having a non-existing value'.

Some Types (there are more!)

Type	Description	Range
char	Usually a byte character.	-128 to 127
unsigned char	A byte	0 to 255
short, int	At least a 16-bit integer	–32,768 to 32,767
unsigned short, unsigned int	If 16-bit → a word.	0 to 65,535
float	Single precision floating point (4 bytes).	3.4E +/- 38 (7 digits)
double	Double precision floating point (8 bytes).	1.7E +/- 308 (15 digits)

Note! Implementation dependent.

Note! Limits are defined in limits.h.

ASCII

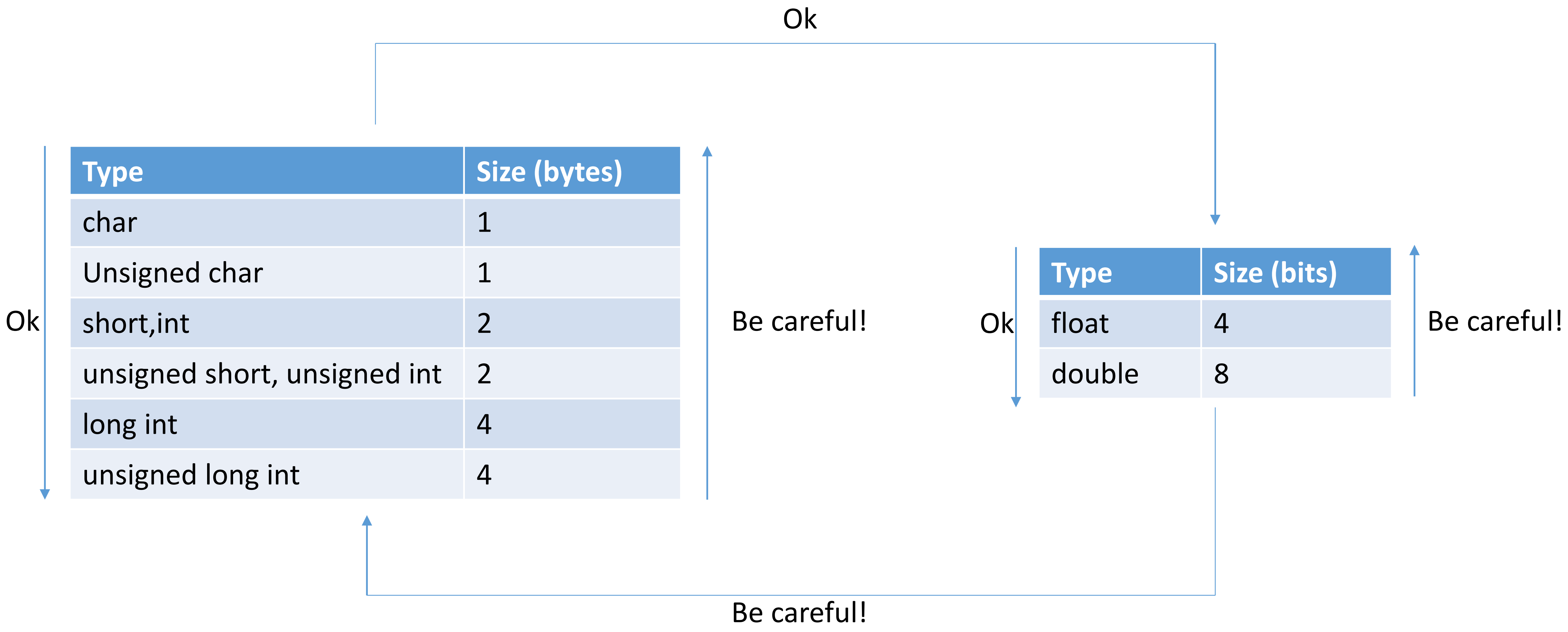
- char code value (an internal representation)
- An integer (byte)
- 0-255

Value	Character	Control
10	-	LF (Line Feed)
12	-	CR (Carriage Return)
65	A	
97	a	

Casting

- Converting from one type to another.
- Implicit/automatic
  - Different operands, assignment, function arguments, return value from a function.
- Explicit
  - (type)expression

Casting



Ok

Type	Size (bytes)
char	1
Unsigned char	1
short,int	2
unsigned short, unsigned int	2
long int	4
unsigned long int	4

Ok

Be careful!

Type	Size (bits)
float	4
double	8

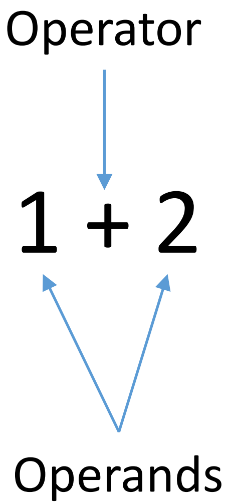
Ok

Be careful!

Be careful!

Operators

Operand vs. Operator



Operator

1 + 2

Operands

Expressions

- They return values (statemens don’t)

Assignment

Operator	Meaning	Comment
=	Assign	a=2
+=	Add and assign	a+=2 → a=a+2
-=	Subtract and assign	a-=2 → a=a-2
*=	Multiply and assign	a*=2 → a=a*2
/=	Divide and assign	a/=2 → a=a/2

There are more assignment operators!

Arithmetic

Operator	Meaning	Comment
+	Add	1+2
-	Subtract	1-2
*	Multiply	1*2
/	Divide	9.0/2 → 4.5 9/2 → 4 Note! If operands are integers → Integer division!
%	Modulo	4 % 2

Precedence

Order	Operator	Comment
1	()	Force order of evaluation.
2	* / %	Left to right.
3	+ -	Left to right.

$$y = \frac{2x - 1}{5}$$ → y=(2x-1)/5

Increment/Decrement (Unary)

- ++ → increment
- -- → decrement
- Prefix
  - Increment/decrement before its value is used
- Postfix
  - Increment/decrement after its value is used

Boolean

- _Bool
- Is an integer.
- 0 → False
- Everything else → true
- Standard dependent
  - Exists as a type in newer versions (bool).

Relational

Operator	Meaning	Comment
<	Less than	1<2 → 1 (true)
<=	Less than or equal	1<=2 → 1 (true) 2<=2 → 1 (true)
>	Greater than	1 > 2 → 0 (false)
>=	Greater than or equal	1 >= 2 → 1 (true) 2 >= 2 → 1 (true)
==	Is equal to	1 == 1 → 1 (true) 1 == 0 → 0 (false)
!=	Not equal to	1 != 1 → 0 (false) 1 != 0 → 1 (true)

Be Warned! Testing for equality (or not) when operands are of floating point type is not recommended !!

Floating points and Equality

- Problem since floats (sometimes) aren't represented exactly.
- Use epsilons when testing for equality !
  - FLT_EPSILON
  - DBL_EPLSILON

Include float.h

Example

Logical (not Bitwise!)

Operator	Meaning	Comment
&&	And	1 && 0 → 0
||	Or	1 || 0 → 1 1 || 2 → 1
!	Not	!1 → 0 !0 → 1

Remember

0 → False

Anything else → True

Logical Operators and Short Circuiting

- Logical operators are read *from left to right*.
- If an operator to the left is false, then the operators to the right won't be evaluated (e.g. if they are functions!).

```
int a = 1, b = 0, c = 0;

If( c && (a || b))
{
  // Some code
}
```

| | will not be evaluated!

Example

- Create a program that allows a user to add two integers. The sum is printed in the screen.