1DT301 - C Programming

Lecture 2

Linnaeus University, 2017

[OPNOVA]

ENGINEERED INNOVATION

Asm / C Comparison

(Blinking LED on an ATmega328p!)

```asm
.org 0x0000
    jmp main

main:
;------- Setup stack
ldi r16, high(RAMEND);
out SPH,r16
ldi r16, low(RAMEND)
out SPL,r16

;---------
sbi DDRB, 5         ; Set Arduino Digital
Pin 13 to OUTPUT
sbi PORTB, 5   ; LED ON

loop:
sbi PORTB, 5   ; LED ON
rcall Delay200ms
cbi PORTB, 5   ; LED OFF
rcall Delay200ms
rjmp loop
```

```c
// ATmega328p
#define F_CPU 16000000UL

#include <avr/io.h>
#include <util/delay.h>

void main(void)
{
     DDRB = 1 << DDB5;

    while (1)
    {
          PORTB = PORTB | (1 << PORTB5);
          _delay_ms(200);
          PORTB = PORTB & ~(1 << PORTB5);
          _delay_ms(200);
    }
}
```

Control Flow

Control Flow

- Determines in what way your program flows.
- If the sun is shining and the temperature is more than 20 °C then go bathing, otherwise go sleeping at home.

if-else

false true

```
if(logical expression)
   statement1;
else
   statement2;
```

```
if((sun is shining) && (temp > 20))
    Go bathing;
else
    Go sleeping at home;
```

What if more statements are needed ?

```
if(logical expression)
   statement1;
   statement2;
else
   Statement3;
   Statement4;
```

Compiler error !

?

Example!

else-if

false true

```
if(logical expression 1)
{
   Statement1;
}
```

false true

```
else if (logical expression 2)
{
   Statement2;
}

else
{
   Statement3;
}
```

Switch

```
switch(expression)
{
   case constant1:
    break;
   case constant2:
    break;
   case constant_n:
    break;
   default:
}
```

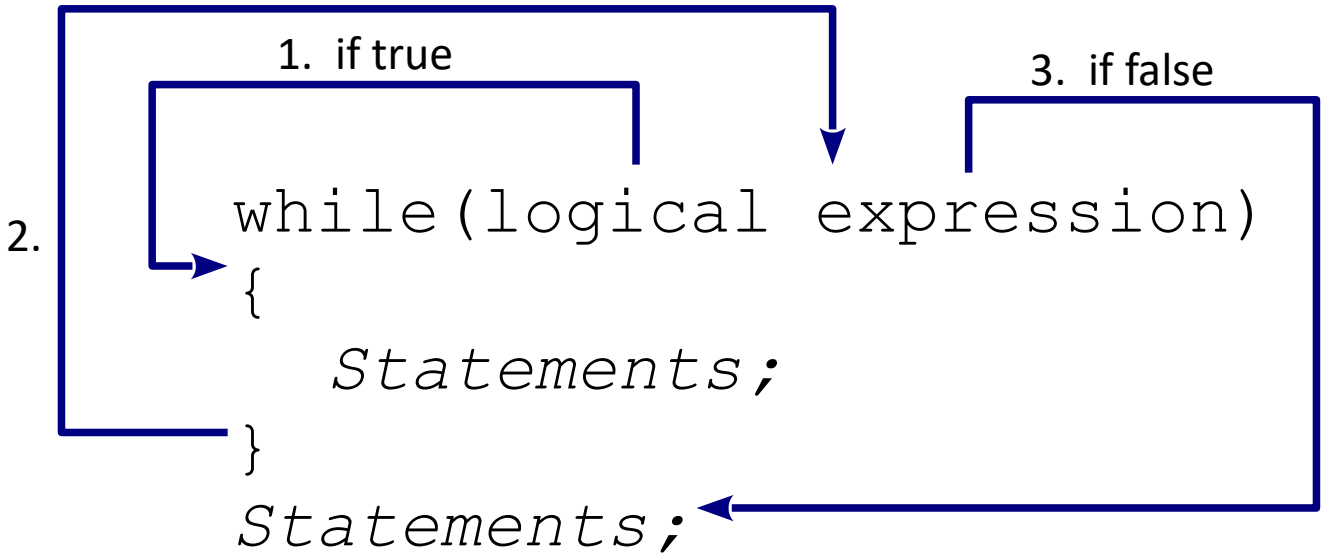
`break` can be omitted → new meaning!

Loops

- Used for repeating execution.
- There need to be a logical expression telling when to stop repeating. If not, the loop will go on forever.
- while, for, do-while
- goto + labels
  - Considered to be bad[1].
  - Compare to ASM

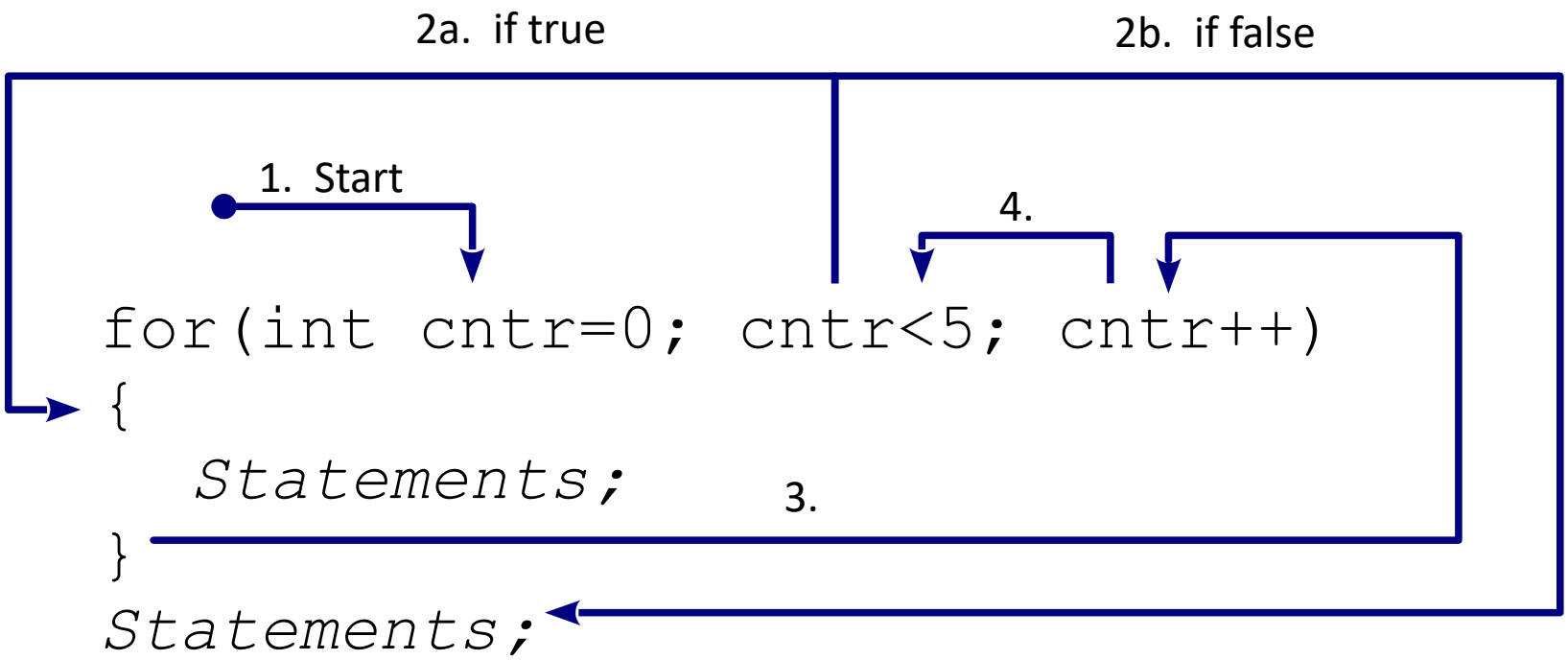
[1] See the classic paper "Go To Considered Harmful" by Edsger W. Dijkstra (http://www.u.arizona.edu/~rubinson/copyright_violations/Go_To_Considered_Harmful.html).

while



Executed zero or more times.

for (typical)



Executed zero or more times.

for (more formal)

```
for(init expr;logical expr;loop expr)
{
   Statements;
}
Statements;
```

for ↔ while

```
for(int cntr=0; cntr<5; cntr++)
{
   Statements;
}
Statements;
```

```
int cntr=0;
while(cntr<5)
{
   Statements;
   cntr++;
}
Statements;
```

break/continue

```
int cntr=0;
while(cntr<5)
{
   Statements;

   if(cntr % 2 == 0)
     break;

   if(cntr % 3 == 0)
     continue;

   Statements;
   cntr++;
}
Statements;
```

Works with all loop constructs!

Pointers

Intro

Pointer

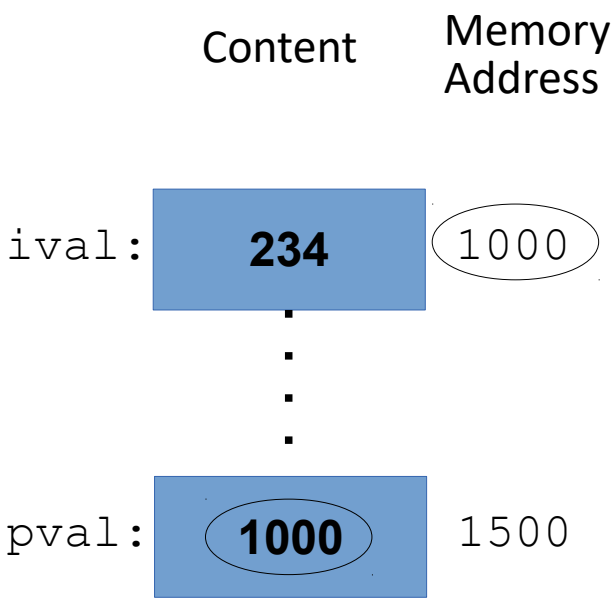
- Variable referring to a memory location.
- Useful when, for instance,
  - Allocating memory
  - Manipulating function arguments.
  - Pointing to different variables
- This is the same thing as in Assembler but the notation is different!

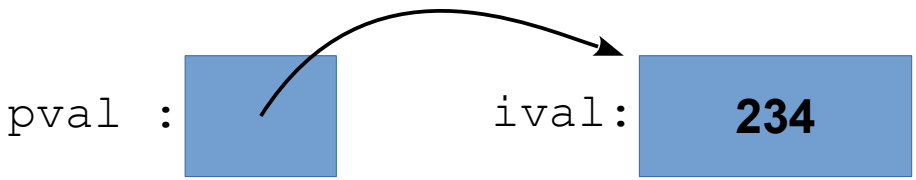
Pointers

- Declaring
  - `type *name;`
- Getting an address of a variable
  - Put & character in front of the variable.

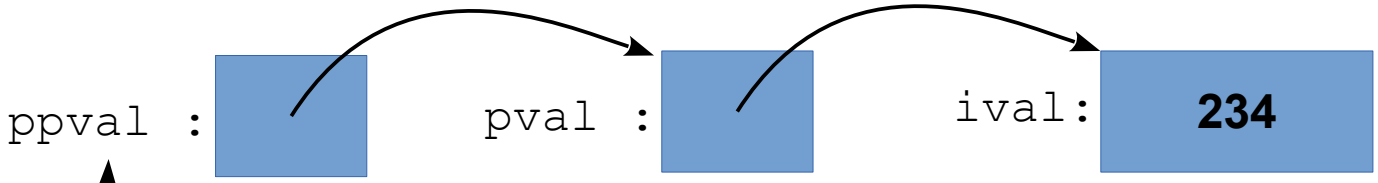
Pointer

```
int ival = 234;
int *pval = &ival;
```





Pointer



Declared as and assigned to ?

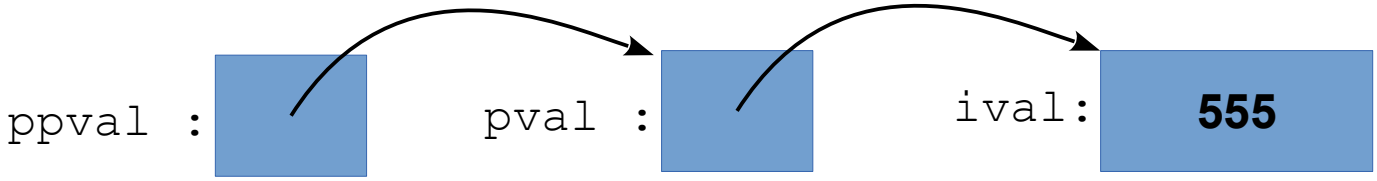
```
int **ppval = &pval;
```

Pointer

```
*pval = 555;
```

What is the value of ival ?

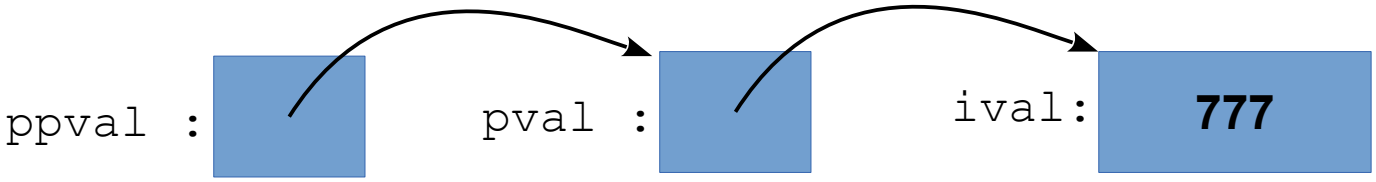
Read as : Wherever `pval` points, assign 555.



Pointer

```
ival = 777;
   printf("%i", **ppval);
```

What is printed?



Example

That C program again

```
// ATmega328p
#define F_CPU 16000000UL

#include <avr/io.h>
#include <util/delay.h>

void main(void)
{
    DDRB = 1 << DDB5;

    while (1)
    {
        PORTB = PORTB | (1 << PORTB5);
        _delay_ms(200);
        PORTB = PORTB & ~(1 << PORTB5);
        _delay_ms(200);
    }
}
```

DDRB and PORTB are dereferenced pointers!

Dynamic memory

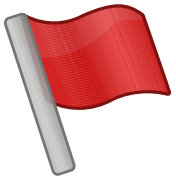
Why ?

- Static memory requires that You know from the beginning how much You need. Seldom the case.

Allocate/deallocate

- Requesting chunks from ”OS” (bytes).
- What is allocated, has to be removed.
- Tricky to find a good method for securing deallocation
  - E.g the one who creates destroys.
  - 'Owner' of the memory
- In Assembler → Write Your own memory manager!

Garbage collector ?

- **NO!** 
- If no one removes allocated memory *manually*, it will remain in memory.
- Space hog.

Allocate

- `void *malloc(size_t how_many_bytes)`
  - size is in bytes
  - return a pointer to the allocated memory
- casting `void *` to the appropriate type.
-
```
void *memset(void* from_where,
             int set_sto_what,
             size_t how_many_bytes)
```

Deallocate

- `void free(void *ptr)`
  - ptr points to the chunk being deallocated.
- Don't forget to do this!