# Metropolia University of Applied Sciences

Programming

TI00AA43-3002

Lecture 2

Sami Sainio

sami.sainio@metropolia.fi

# Plan for going forward

- Variables and printing to screen
- **If, else, while and for loops**
- I/O
- Functions and tables
- File handling
- Pointers and arrays
- Simple structures
- Program structure and design

# 2. Lectures

If, else, while and for loops

# Control flow

- When creating programs, it is usually necessary to control the flow of the application
  - Examples
    - Programmer wants to repeatedly print out a group of results from calculations like when converting temperature from celcuis to kelvin from a given range
    - Taking actions depending on what the user inputs, if user inputs A action X will be taken and if user inputs B action Y will be taken

```
#include <stdio.h>

int main(void)
{
        float kelvin, celsius;
        int lower, upper, step;

        lower = 0;          // Lowest temperature in Kelvins
        upper = 300;        // Highest temperature in Celsius
        step = 30;          // Conversion ”step”

        kelvin = lower;
        celsius = (kelvin - 273.15);

        while (celsius <= upper+30) {
                printf("%4.0f %4.2f\n", kelvin, celsius);
                kelvin = kelvin + step;
                celsius = (kelvin - 273.15);
        }
        return 0;
}
```

Float variables

Printing and conversion in while loop

Make printing look clear

# while-loop

```
while(round<100){
        round++;
}
```

- Code between the { } will be executed as long as the statement in ( ) is true

```
#include <stdio.h>

int main(void)
{
        float kelvin, celsius;
        int lower, upper, step;

        lower = 0;          // Lowest temperature in Kelvins
        upper = 300;        // Highest temperature in Celsius
        step = 30;          // Conversion ”step”

        kelvin = lower;
        celsius = (kelvin - 273.15);

        for (kelvin = lower; celsius <= upper; kelvin = kelvin + step) {
                celsius = (kelvin - 273.15);
                printf("%4.0f %4.2f\n", kelvin, celsius);
        }
        return 0;
}
```

Float variables

Printing and conversion in for loop

Make printing look clear

# for-loop

```
for(i=0; i<10; i++){
        printf(”%d round number”, i);
}
```

- i=0 -> declare i to 0
- i<10 -> condition for execution
- i++ increase the value of i after each round

```
#include <stdio.h>

int main(void)
{
        float kelvin, celsius;
        int lower, upper, step;

        lower = 0;          // Lowest temperature in Kelvins
        upper = 300;        // Highest temperature in Celsius
        step = 30;          // Conversion ”step”

        kelvin = lower;
        celsius = (kelvin - 273.15);

        do {
                printf("%4.0f %4.2f\n", kelvin, celsius);
                kelvin = kelvin + step;
                celsius = (kelvin - 273.15);
        } while(celsius <= upper+30);
        return 0;
}
```

Float variables

Printing and conversion in do-while loop

Make printing look clear

# do-while loop

```
do {
        printf(”round %d\n”, i);
        i++;
    } while(i < 10);
```

- Code between { } is executed as long as the condition between ( ) is true
- Condition is checked after the code is ran

# If else and operators

```
if(cha == '\n' || cha == ' ')
{
    printf("character was a newline or whitespace");
}
else
{
    printf("character was something else");
}
```

- Operators that are available:
  - || = logical OR, && = logical AND

# I/O (input/output, reading/printing)

- Stdio models character and text input/output (I/O)
  - I/O is a stream of characters that are on one or multiple lines
- Reading / writing characters one character at a time
  - cha = getchar(); // int cha;
  - putchar(cha); // int cha;
- getchar() return either ASCII code from the character read or constant EOF (defined in stdio.h)
- EOF marks the end of the stream

# ASCII table

| Dec | Hex | Name | Char | Ctrl-char | Dec | Hex | Char | Dec | Hex | Char | Dec | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | Null | NUL | CTRL-@ | 32 | 20 | Space | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | Start of heading | SOH | CTRL-A | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | Start of text | STX | CTRL-B | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | End of text | ETX | CTRL-C | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | End of xmit | EOT | CTRL-D | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | Enquiry | ENQ | CTRL-E | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | Acknowledge | ACK | CTRL-F | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | Bell | BEL | CTRL-G | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | Backspace | BS | CTRL-H | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | Horizontal tab | HT | CTRL-I | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | 0A | Line feed | LF | CTRL-J | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | 0B | Vertical tab | VT | CTRL-K | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | 0C | Form feed | FF | CTRL-L | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | 0D | Carriage feed | CR | CTRL-M | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | 0E | Shift out | SO | CTRL-N | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | 0F | Shift in | SI | CTRL-O | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | Data line escape | DLE | CTRL-P | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | Device control 1 | DC1 | CTRL-Q | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | Device control 2 | DC2 | CTRL-R | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | Device control 3 | DC3 | CTRL-S | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | Device control 4 | DC4 | CTRL-T | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | Neg acknowledge | NAK | CTRL-U | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | Synchronous idle | SYN | CTRL-V | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | End of xmit block | ETB | CTRL-W | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | Cancel | CAN | CTRL-X | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | End of medium | EM | CTRL-Y | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | Substitute | SUB | CTRL-Z | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | Escape | ESC | CTRL-[ | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | File separator | FS | CTRL-\ | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | Group separator | GS | CTRL-] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | Record separator | RS | CTRL-^ | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | Unit separator | US | CTRL-_ | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | DEL |

http://www.commfront.com/ascii-chart-table.htm

# Reading characters 1/3

- Terminal reads one line at a time
- This means that when reading characters you need to press enter in order to receive something at the program
- Notice that pressing enter != EOF!

# Reading characters 2/3

```
#include <stdio.h>
int main(void)
{
    int cha;

    while((cha= getchar()) != EOF)
        putchar(cha);
}
```

Using int type of a variable to save input

!= means not equal

1. Call function getchar()
2. The character that the function returns is saved
3. Check if the character was EOF

# Reading characters 3/3

```
#include <stdio.h>
int main(void)
{
    int age, height;
    printf(“Age and height, separated by space: ");
    scanf("%d %d", &age, &height);
    printf(“Age %d, height %d\n", age, height);
    return 0;
}
```

- scanf(), as printf() but for reading
- Character string as in printf(), but arguments need to be accompanied with & operator (address of the varible) so the value can be saved