Metropolia University of Applied Sciences

Programming
TI00AA43-3003
Lecture 6

Sami Sainio

sami.sainio@metropolia.fi

Metropolia

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

6. Lectures

Simple structures

Structures

- Used to group information (data) under one common name
- This means, that in single structure there can exist several same type of variables and/or collection of different variable types
  - Can be accessed via the structures name
- Can be used in many places, one use example is reading data from file that is then passed to software

```
// file structure.h
struct student
{
int id;
char name[20];
float successpercent;
} register;
```

```
// file structure.c
#include <stdio.h>
#include <string.h>
#include ”structure.h"          /* header file, where structure is
                                   defined */
int main()
{
        register.id=1;
        strcpy(register.name, ”Student");
        register.successpercent = 86.5;

        printf(" Id is: %d \n", register.id);
        printf(" Name is: %s \n", register.name);
        printf(" Success: %f \n", register.successpercent);
        return 0;
}
```

Lets observe the structure in more detail

- The code below will create a structure thats name is ”register” that is of type student
- Structure includes three different variables, that can be accessed via STRUKTURENAME.VARIABLE

```
// file structure.h
struct student
{
		int id;
		char name[20];
		float successpercent;
} register;
```

Lets observe the structure in more detail

- Structure created in .h file and then included in the main code will work as it would have been created in the main
- Accessing as in the previous slide

```
// file structure.c
#include <stdio.h>
#include <string.h>
#include ”structure.h"
int main()
{
            register.id=1;
            strcpy(register.name, ”Student");
            register.successpercent = 86.5;

            printf(" Id is: %d \n", register.id);
            printf(" Name is: %s \n", register.name);
            printf(" Success: %f \n", register.
(doest work)          successpercent);
            return 0;
}
```

Using stuctures

```
struct op{
		int id;
		char name[20];
		float successpercent;
		} r[40];
```

- Code creates an array with 40 cells, where in every index there is one structure
- Access now: r[index].variblename
- Example: passing ID to variable:

```
for(i=0;i<(sizeof(r)/sizeof(r[0]));i++)
{
		r[i].id = i;
}
```

Observations from previous slide

- sizeof() function returns variable size in memory
- Can be used as on previous slide, to find out the length of the array, even tho there are structures in the array cells

Structure definition

- Structure can be defined without the .h file as well:

```
#include <stdio.h>

struct phonenumber{
char name[20];
int number;
};

int main() {
struct phonenumber example;
return 0;
}
```

Structure name is now example