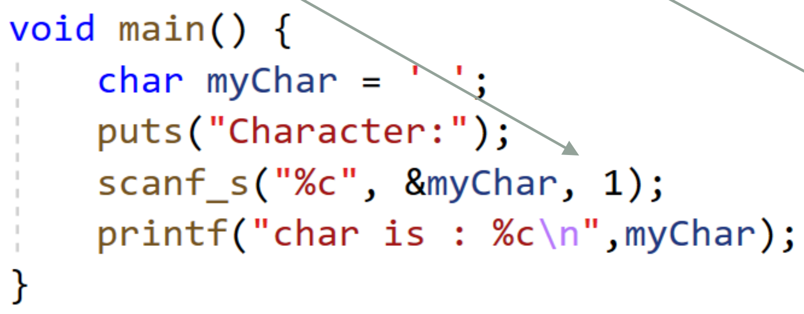
PROGRAMMING

CT103

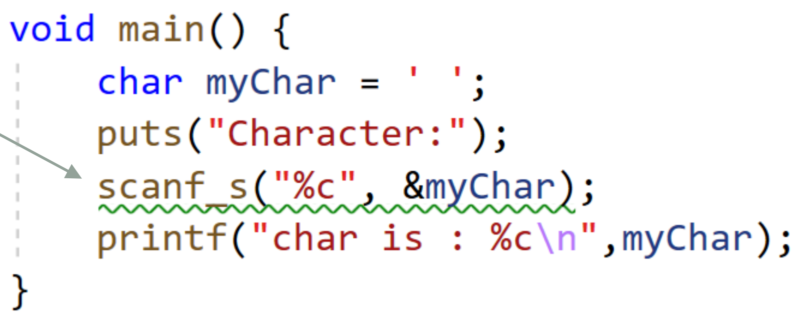
Week 8b

Question from Previous Lecture

- Why this and not this?



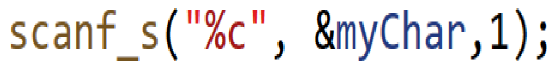
```c
void main() {
    char myChar = ' ';
    puts("Character:");
    scanf_s("%c", &myChar, 1);
    printf("char is : %c\n",myChar);
}
```



```c
void main() {
    char myChar = ' ';
    puts("Character:");
    scanf_s("%c", &myChar);
    printf("char is : %c\n",myChar);
}
```

- Answer: Both are fine and will run.

- However, when scanning characters, scanf_s wants to know how many characters. Hence the “1” in:



```c
scanf_s("%c", &myChar,1);
```

Lecture Content

- Last lecture (Week 8a):
  - Testing characters
  - Character mapping
  - Arrays of strings
  - Example C program

- Today’s evening lecture (Week 8b):
  - Functions
  - Writing functions
  - Functions in C
  - Example C program

FUNCTIONS

Functions

- What is a function?

- **Definition**: A function is a piece of code that can be called whenever we need to execute that code.

Functions

- What functions have we seen so far?

- We have seen many functions in C:
  - strlen()
  - isalpha()
  - isdigit()
  - isupper()
  - islower()
  - isspace()

- These are all examples of pieces of code that we can call in our program to achieve some task.

Functions

- What is the point of functions?

- Benefits:
  - Functions allow us to reuse code, therefore avoid repetition.
  - More readable programs.
  - Enables us to divide complex problems into simpler ones.
  - Easier to make changes to program.

WRITING FUNCTIONS

Function Template

- All functions have the following template:

```
type name (parameters){
        return;
}
```

- Type = data type returned by the function (can be void).
- Name = function name.
- Parameters = data we are giving to the function (can be empty).
- Return = what data is returned by the function (can also return nothing).

Function Type

- Like a variable, a function must have a ***type***
- It can be one of the standard variable types (char, double, float, int) or it can be void
- The type tells the compiler what *type* of variable the function returns
  - For example
  - getchar() returns a char
  - strcmp() returns an int

Why return anything?

- Functions can return a value or answer to some calculation or query
  - E.g. int getEmployeeAge(int employeeID);

- When we have the answer to the calculation or query, we will likely want to use this somewhere else in our program.

- In order to do this, we need to return that value from the function.

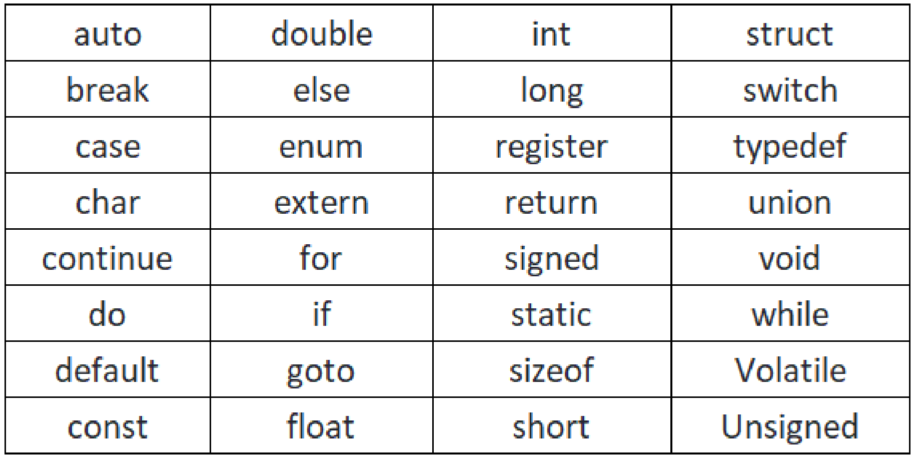
Naming functions

- Function names can’t contain spaces.
- You should give your function a helpful name that reflects what it does.
- Each functions is declared with parentheses “**()**” after the function name (even if it doesn’t don’t take any parameters), e.g. `void main().`
- You can’t name your function using a “reserved word”.

Reserved Words

- You can’t name your function using a “reserved word”.

- What is a reserved word?
  - There are 32 reserved words that have predefined meaning in C. You therefore can’t use these as variable names.



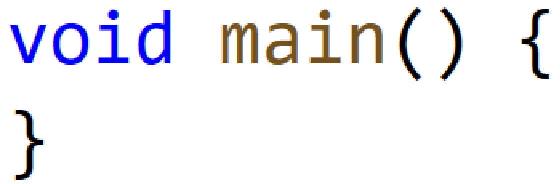
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	Volatile
const	float	short	Unsigned

Function Prototypes

- Before your compiler will let you use a function, you have to give it a prototype.
- We have to do this before we call it, normally before the main() function, and after any #include or #define directive.
- The .h files (header files) contain the prototypes for C library functions we call.

Writing a Function

- We have already actually written a function:



```
void main() {
}
```

- Every time we wrote our C programs, we wrote our code inside of a function **main()**.

main()

- main() is the first function called when a program is executed
- When it is finished the program exits
- Main() can return nothing or an integer

```
int main()
{
  return 0;
}

void main()
{
  return;
}
```

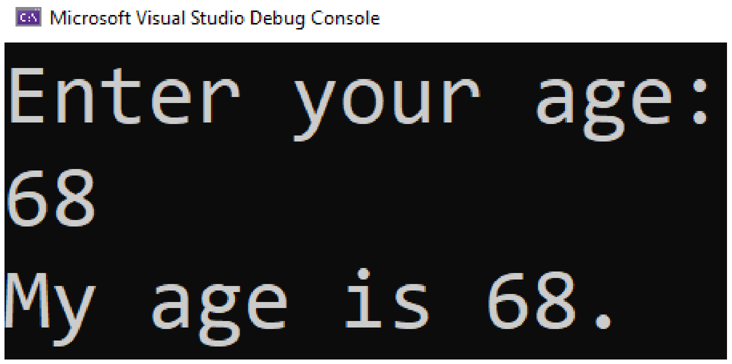
- so… the “type” of a function specifies what it returns (*void* if nothing)

FUNCTIONS IN C

C Program without Function EG1

- Simple C program that reads in an age and prints it to the screen.

```c
void main() {
    int age;
    puts("Enter your age:");
    scanf_s("%d", &age);
    printf("My age is %d.\n", age);
}
```

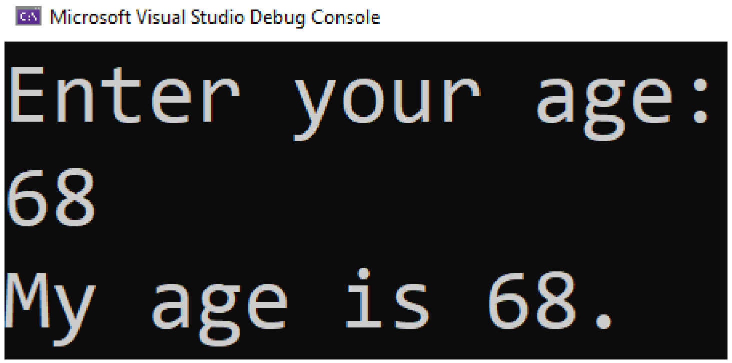


Microsoft Visual Studio Debug Console

```
Enter your age:
68
My age is 68.
```

C Program with Function EG1

- C program that creates a function to read in an age.
- Notice how this function does not read in any parameters.



Microsoft Visual Studio Debug Console

```
Enter your age:
68
My age is 68.
```

```c
#include <string.h>
#include <ctype.h>
#include <stdio.h>

int readAge();

void main() {
    int myAge = readAge();
    printf("My age is %d.\n", myAge);
}

int readAge() {
    int age;
    puts("Enter your age:");
    scanf_s("%d",&age);
    return age;
}
```

Function prototype

Main (we should be familiar with this one)

Function itself

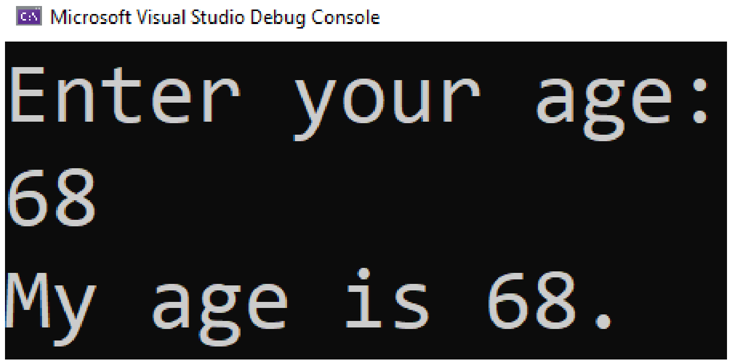
C Program EG1 Comparison

- These programs do the same thing.

No Function

Don't forget header files here.

```c
void main() {
    int age;
    puts("Enter your age:");
    scanf_s("%d", &age);
    printf("My age is %d.\n", age);
}
```



Microsoft Visual Studio Debug Console

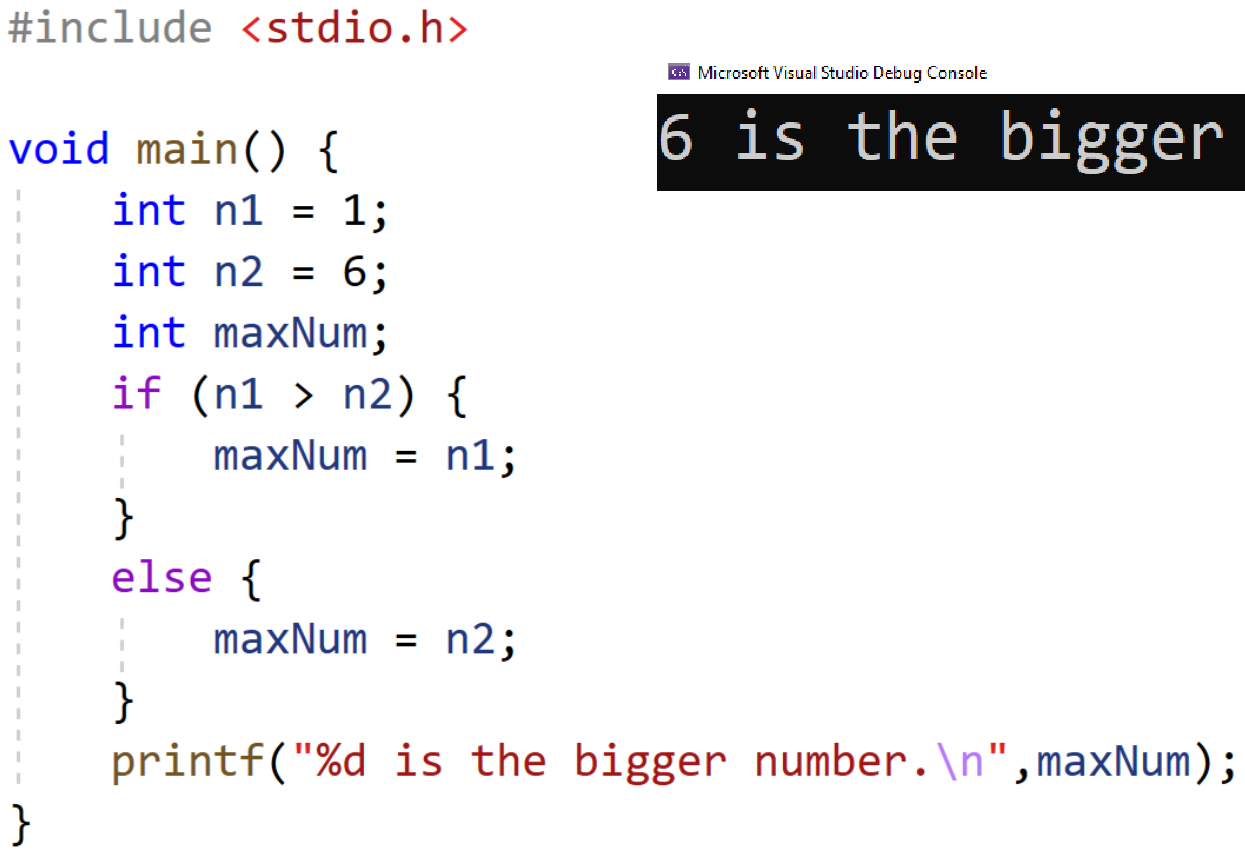
```
Enter your age:
68
My age is 68.
```

Using a Function

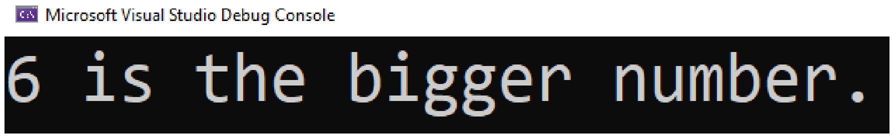
```c
#include <string.h>
#include <ctype.h>
#include <stdio.h>

int readAge();

void main() {
    int myAge = readAge();
    printf("My age is %d.\n", myAge);
}

int readAge() {
    int age;
    puts("Enter your age:");
    scanf_s("%d",&age);
    return age;
}
```

C Program without Function EG2

- Get the max number out of 2 numbers:



```c
#include <stdio.h>

void main() {
    int n1 = 1;
    int n2 = 6;
    int maxNum;
    if (n1 > n2) {
        maxNum = n1;
    }
    else {
        maxNum = n2;
    }
    printf("%d is the bigger number.\n",maxNum);
}
```



Microsoft Visual Studio Debug Console

```
6 is the bigger number.
```

C Program with Function EG2

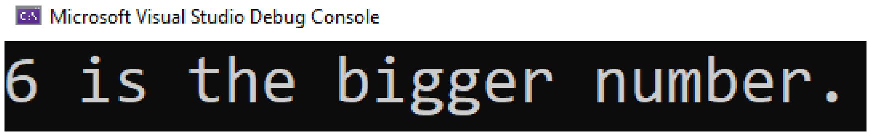
- Get the max number out of 2 numbers:

```c
int maxNums(int num1, int num2);

void main() {
    int maxNum;
    int n1 = 5;
    int n2 = 6;
    maxNum = maxNums(n1, n2);
    printf("%d is the bigger number.\n",maxNum);
}

int maxNums(int num1, int num2) {
    if (num1>num2) {
        return num1;
    }
    else {
        return num2;
    }
}
```

Function prototype

Main (we should be familiar with this one)

Function itself



```
6 is the bigger number.
```

C Program EG2 Comparison

- If these programs do the same thing, why would you use functions? This program is much longer…

No Function

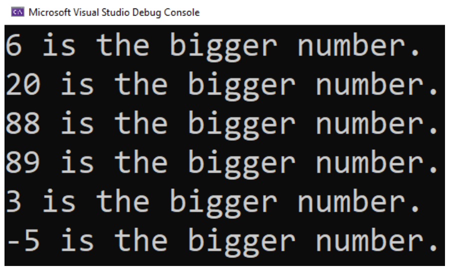
```c
#include <stdio.h>

void main() {
    int n1 = 1;
    int n2 = 6;
    int maxNum;
    if (n1 > n2) {
        maxNum = n1;
    }
    else {
        maxNum = n2;
    }
    printf("%d is the bigger number.\n",maxNum);
}
```

Using a Function

```c
int maxNums(int num1, int num2);

void main() {
    int maxNum;
    int n1 = 5;
    int n2 = 6;
    maxNum = maxNums(n1, n2);
    printf("%d is the bigger number.\n",maxNum);
}

int maxNums(int num1, int num2) {
    if (num1>num2) {
        return num1;
    }
    else {
        return num2;
    }
}
```

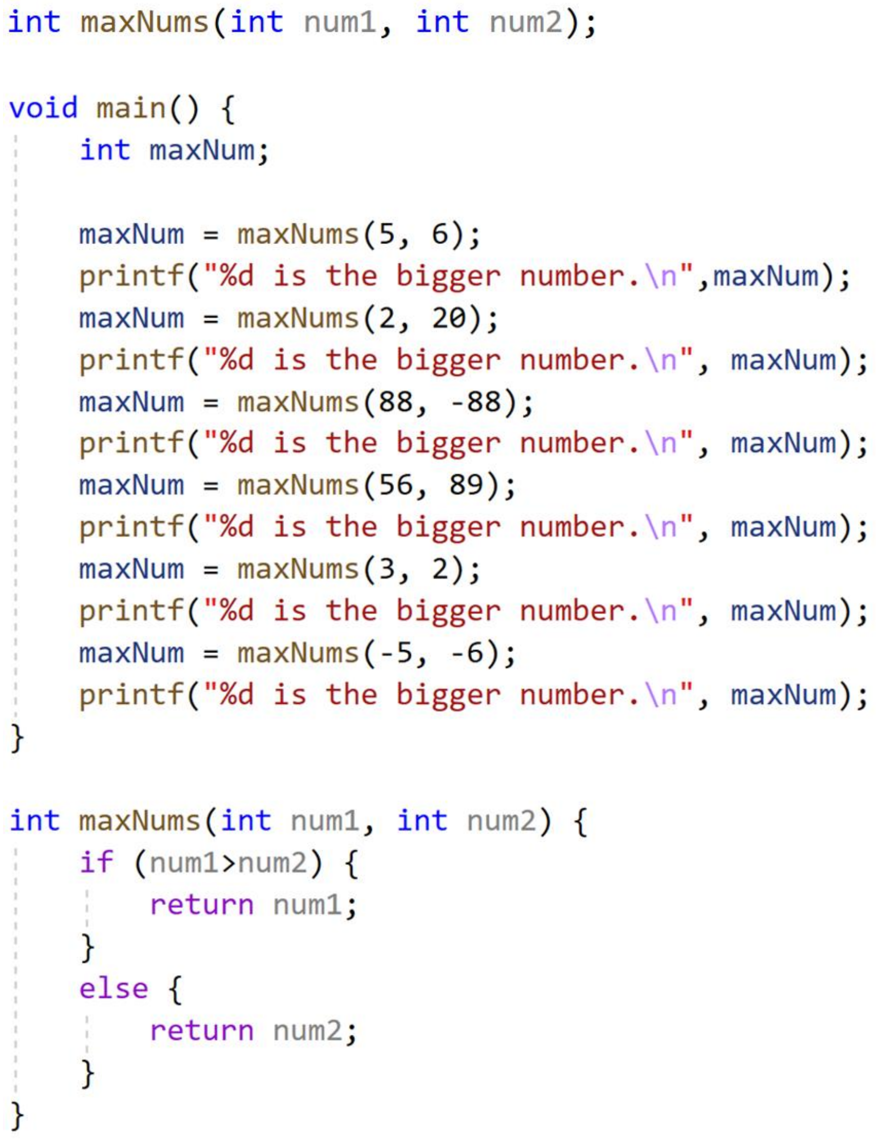
C Program with Function Cont.

- Well what if I wanted to do more than 1 comparison?
- If I use a function, I can simply call the function again.
- This is much more scalable than not using a function.



Microsoft Visual Studio Debug Console

```
6 is the bigger number.
20 is the bigger number.
88 is the bigger number.
89 is the bigger number.
3 is the bigger number.
-5 is the bigger number.
```



```c
int maxNums(int num1, int num2);

void main() {
    int maxNum;

    maxNum = maxNums(5, 6);
    printf("%d is the bigger number.\n",maxNum);
    maxNum = maxNums(2, 20);
    printf("%d is the bigger number.\n", maxNum);
    maxNum = maxNums(88, -88);
    printf("%d is the bigger number.\n", maxNum);
    maxNum = maxNums(56, 89);
    printf("%d is the bigger number.\n", maxNum);
    maxNum = maxNums(3, 2);
    printf("%d is the bigger number.\n", maxNum);
    maxNum = maxNums(-5, -6);
    printf("%d is the bigger number.\n", maxNum);
}

int maxNums(int num1, int num2) {
    if (num1>num2) {
        return num1;
    }
    else {
        return num2;
    }
}
```

EXAMPLE PROBLEM

Salary Tax Function Problem

- You are writing software to process employees salaries:
  - Write a function called “readSalary”.
  - readSalary does not return anything.
  - This function should read in a tax threshold in Euro as a parameter.
  - The function should ask the user to enter the employee salary.
  - The function should then check if the salary is >, <, or = the tax threshold.
  - You should print a message to the console saying which of these is the situation.
  - Test your function by passing in the value of €44,000 as a tax threshold when you call the readSalary function in main.

Salary Tax Function Problem

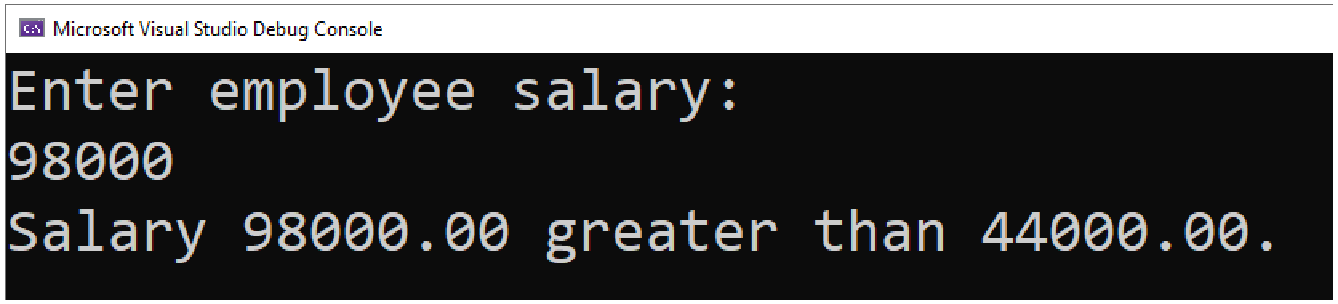
- Go to C program solution.

Salary Tax Function Problem

```c
#include <string.h>
#include <ctype.h>
#include <stdio.h>

void readSalary(float taxT);

void main() {
    readSalary(44000);
}

void readSalary(float taxT) {
    float salary;
    puts("Enter employee salary:");
    scanf_s("%f",&salary);

    if (salary> taxT) {
        printf("Salary %0.2f greater than %0.2f.\n",salary, taxT);
    }
    else if (salary< taxT) {
        printf("Salary %0.2f less than %0.2f.\n", salary, taxT);
    }
    else {
        printf("Salary %0.2f = %0.2f.\n", salary, taxT);
    }
}
```

Salary Tax Function Problem

- C Program Output:


Microsoft Visual Studio Debug Console

```
Enter employee salary:
98000
Salary 98000.00 greater than 44000.00.
```