Overview course Programming 1

1. Variables 1
2. Variables 2
3. Variables 3
4. Conditionals
5. Iterations
6. Functions 1
7. Functions 2
8. Arrays
9. Strings – Game
10. Classes 1 – Encapsulation
11. Classes 2 – Static const

Topics

- Compound statements
- Scope and lifespan of variables

(simple) Statements

- A statement is an expression followed by a ';'
- Examples:
  - `int number {0};`
  - `number += 5;`
  - `number = 145 / 8745;`
  - `4 + 9;`
  - `;       // ; -> is a null statement (!)`

See also: [Statements on MSDN]

Compound-Statement or block

- A block is a group of statements
  - separated by semicolons (;)
  - grouped together in a block enclosed in curly braces: { }
- { [statement-list] }

Compound-Statement or block

- Example:

```
{
    int number{10};
    number += (14 + 5);
}
```

- never used in this form

Scope and lifespan

- Scope: *where* can I use my variables?
- Lifespan: *when* are my variables created, and when are they removed?

Kinds of variables

- The scope and lifespan of a variable depends on what kind of variable it is:

1) Global variable: variables that are *declared* outside of any structure or function {}.
2) Local variables: variables that are declared *inside a function* or *inside a structure {}*.
3) Other: see later

Scope of a variable

- Scope: where can I **use** my variables:

1) A global variable can be used from any function and from any structure inside the file where the variable was declared (for now: anywhere).
2) A local variable can only be used in the function where it was declared, or in the block enclosed in braces {} where it was declared.

Lifespan

- ➢ A global variable:
- Is created when the program starts.
- Is removed when the program ends.

Lifespan

- A local variable:
  - Is created on the line where the variable is declared.
  - Is automatically removed
    - at the end of the function where the variable was created
    - at the end of the block where the variable was created

Variables with the same name

Possible situations:

- Two local variables with the same name in the same function and block?
- Two local variables with the same name but in different functions or block?

Variables with the same name

Possible situations:

- ~~Two local variables with the same name in the same function and block?~~
- Two local variables with the same name but in different functions or block?
- Hiding: When two variables have the same identifier (name), the one with a smaller scope hides the variable with the larger scope.

Variables with the same name

```
int i{40}, sum{};
for(int i{0} ; i < 10; ++i)
{
  sum += i;
}
std::cout << i << ‘\n’;
```

// This prints : ?

Variables with the same name

```
int i{40}, sum{};
for(int i{0} ; i < 10; ++i)
{
  sum += i;
}
std::cout << i << ‘\n’;
```

// This prints : 40 (!)

Tip:

- Avoid a wide scope.
- Do only use global variables if
  - the variable is needed in more than one function
  - the value of the variable must be retained when used in a function.
- Why? Readability, maintainability