OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Architecting and Engineering
Main Memory Database Systems in Modern C
Database and Software Engineering Working Group

Magdeburg, Dec 01st 2017

Marcus Pinnecke, M.Sc.
Prof. Dr. Gunter Saake

Winter Term 2017/2018

Storage Classes & Expressions

submission deadline: Dec. 08th 2017 11:59pm
Note: this is a one-week exercise sheet

This is a *per-student exercise* sheet, i.e., you are not allowed to submit a group solution.

You must submit your solution in time via a pull request to the lectures exercise Git repository[1] (cf. sheet No. 1). For this, branch from the official branch `master`, and commit your solutions in a directory `sheet_05/<your last name>`. For your pull request, use the remote branch `submissions` as target for merging.

Prepare to present details of your solution during the tutorium.

Exercise sheet No. 5

After working on this sheet you will have learned the following

- ✓ Identifier referencing in C programs
- ✓ Identify different operator types in C
- ✓ Conditional expressions in C
- ✓ Storage classes, duration and linkage
- ✓ Operators (and precedence)

This sheet consists of the following tasks:

Task 1	**Scoping**	3 Points
Task 2	**Storage Classes**	3 Points
Task 3	**Expressions and More: Right or Wrong**	2 Points
Bonus	**C Operator Precedence**	+16 Points

Good Luck!

[1] https://github.com/Arcade-Lecture/exercises.git

Task 1 Scoping 3 Points

Consider the following C program:

```
01:  unsigned long a, b, d;
02:
03:  int main(int a, char **c) {
04:      d = (a += 1), (d = 112);
05:      b = a = 23,
06:          b = 42,
07:          b += (a++ + b);
08:      {
09:          int a = b + 5;
10:          b = a;
11:      }
12:      return b - d;
13:  }
```

(1) Does this program terminate successfully?

(2) Image you run this program starting at line 04 until it finished at line 12. For all lines:

a. state the name of the operator(s) that is (are) used!

Example: *04: basic assignment, comma operator, addition assignment*

b. for each operator in (a), state for each operand:

i. the line number to which the identifier refers!

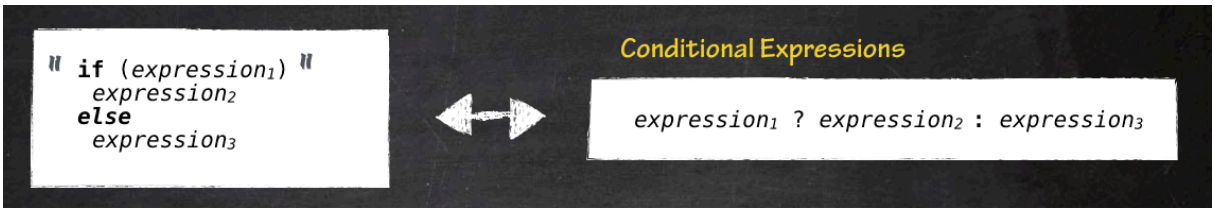
ii. the scope type in which the referred identifier is!

Example: *a: 03 (function argument list scope), d:01 (file scope)*

Provide a textual solution formatted in Markdown in *<repo>*/sheet_05/*<your last name>*/task_1:

Task 2 Storage Classes 3 Points

Conditional expression: A conditional expression is one way to express a conditional choice.



If *expression*$_1$ evaluates to non-zero, then the value of *expression*$_2$ is taken. The value of *expression*$_3$ is taken otherwise.

Consider the following C program:

```
01:  long unsigned long state(int x, register int y)
02:  {
03:      static int state;
04:      state = x > 0 ? x : state;
05:      return x + y;
06:  }
07:
08:  int main(void)
09:  {
10:      auto int a = 23;
11:      short int b = state(a, 42);
12:      register short c = (short) state(state(1, a), state(0, b));
13:      return (c - b - a);
14:  }
```

(1) Does this program terminate successfully?
(2) Give a statement on the storage class, storage duration and linkage of the following variables:
   a. a
   b. b
   c. c
   d. x
   e. y
(3) Image you alter the line `10: auto int a = 23;` as follows:
   I. `10: auto extern int a = 23;`
   II. `10: int a = 23;`

   for (I) and (II) answer the following questions!
      a. Is the altered line allowed in C?
      b. In case (i) yields yes, how does the linkage change?

(4) The function `state` uses a variable `state` that is marked as **`static`**.
   a. What is the effect of **`static`** in this context?
   b. What changes if you remove **`static`**?
   c. Does the result of the function also change when you remove **`static`** and why?

(5) Is the explicit type cast in line 12 necessary?
(6) What happens when calling `state(-42, -42)`? Suggest a solution to solve the issue!

Provide a textual solution formatted in Markdown in *<repo>*/sheet_05/*<your last name>*/task_2:

Task 3 **Expressions and More: Right or Wrong** 2 Points

For each of the following statements, state if the statement is it right or wrong, and give a short explanation why it is right or wrong. Create a text file formatted in Markdown containing your solution. Save your solution to *<repo>*/sheet_05/*<your last name>*/task_3.

Id	Statement	Right	Wrong	Reason
1	The postfix increment operator ++ evaluates to the value `a` for `a++`.			
2	The expression `--a` is equivalent to `a -= 1`.			
3	If `foo` is a function name, then the expression `foo()` results in a pointer to the function `foo`.			
4	Assume `a`,`b` as lvalues of number type, then `(i + j)` is a rvalue.			
5	The line **`register int`** `x` forces to compiler to store `x` in a CPU register.			
6	The storage class *static* leads to automatic storage duration and internal linkage.			
7	A memory leak may occur when allocated storage duration is used.			
8	Given two structs **`struct`** `s1`, and **`struct`** `s2` in the *same* scope. C disallows to assign the same identifier `x` to members in `s1` and `s2` (`i.e.`, `s1.x` and `s2.x` is always invalid)			
9	The type cast operator (*`type`*) has the same precedence as prefix increment operator in C.			
10	The expression `a << b` results *true* iff `a` is less but not equal to `b`.			

Bonus **C Operator Precedence** +16 Point

Assume the declaration of the following variables in an accessible scope:

```
int x, y, z;
```

Consider the following expression

```
z & x ++ - (char) ! - z -- - (x + 5 >> y)
```

Add round enclosing braces such that it reflects the evaluation order according the operator precedence in C! For all operators at the same precedence, assume left to right evaluation.

Example:

```
5 + 3 * 2 * ( 5 - 3 )    →    (5 + ((3 * 2) * ( 5 - 3 )))
```