ReVision Goals

Olaf Bernstein

June 3, 2018

The Goal of this project is the complete revision of my favourite concepts of programming languages. It will be primarily oriented around C++ concepts because its currently one of my favourite and most used programming languages.

Contents

1 Why Not C++?

So why not C++, well C++ might support a grate amount of features, heck you can even make thinks like an NES emulator at compile time, but it has gotten pretty messy lately. For Example, there are still trying to keep the C backwards compatibility, but have extremely many new features. Some serve the same purpose of the C equivalent, but are only there to support the new C++ features.

2 Control Flow

2.1 *while*

1. If condition is true jmp 3.
2. Execute code block.
3. jmp 1.
4. Continue execution.

Syntax:

```
while(condition)
{
        // Some code ...
}
```

2.2 *jmp*

Will jump to the Lable that's specified.

```
LABLE:
// Some code ...
jmp LABLE;
```

You can also insert a optional condition.

```
LABLE:
// Some code ...
jmp if(condition) LABLE;
```

2.3 *for*

1. Execute initialization code. 2. If condition is false jmp 5. 3. Execute code. 4. Execute iterate code. 5. Continue execution.

```
for(initialization; condition; iterate)
{
        // Some code ...
}
```

2.4 *switch*

Generates a jump table that jumps to the cases where the value is the same as the variable value.

```
switch(variable)
{
        case value_1:
        {
                // Some code ...
        }
        case value_2, value_3:
        // Some code ...

        default:
        // Some code ...
}
```

2.5 *if*

1. If the condition is true execute next code block. 2. Optional else code block get executed if 1 is false.
Is also possible to stack if statements using else if's.

```
if(condition_1)
{
        // Some code ...
}
else if(condition_2)
{
        // Some code ...
}
else
{
            // Some code ...
}
```

2.6 *do*

1. Execute code block.
2. If condition is true jmp 1.
4. Continue execution.

Syntax:

```
do
{
            // Some code ...
} while(condition);
```

2.7 *asm*

The code block after the asm keyword will be executed as assembly code.

```
asm
{
            mov eax, 2 ; ...
};
```

3 Composite Types

3.1 *class*

3.2 *enum*

3.3 *uniom*

3.4 *string*

4 Type Modifier

4.1 *const*

4.2 '*[]*'

C Array

4.3 *'[]!'*

Vector

4.4 *'*'*

Pointer

4.5 *'*!'*

Unique Pointer

5 Primitive Types

5.1 Integer

signed	unsigned	Description
int8	*uint8*	8-Bit integer value.
int16	*uint16*	16-Bit integer value.
int32	*uint32*	32-Bit integer value.
int64	*uint64*	64-Bit integer value.

5.2 Floating Point

Type	Description
float	IEEE-32 bit Floating Point.
double	IEEE-64 bit Floating Point.

5.3 Other

Type	Description
char	8-Bit character holding type.
auto	Deduces the type automatic.

6 Operators

6.1 *new*

6.2 *delete*

7 Scope

7.1 *namespace*

7.2 *use*

The scope of the code block gets reduced to the variables in the capture list.

```
use x, y // Capture list
{
        // Some code ...
};
```

Optional you can return a variable just like in a function.

```
z = use x, y // Capture list
{
        // Some code ...
        return val;
};
```