Variables & Memory

Prof. Dr.-Ing. Angela Brennecke
Audio & Interactive Media Technologies
Film University Babelsberg KONRAD WOLF

Variables & Memory

- Whenever you write a computer program and declare variables and functions, the computer allocates memory
- The allocated memory is needed
  - to store any variable and object values
  - to store any function parameter values

Variables & Memory

- The variable's data type is used by the compiler
  - to interpret (read & write) the allocated memory and
  - to reserve the required amount of space in memory

- The variable name is used to
  - associate the piece of memory with the variable in code
  - access and change the variable values stored in memory

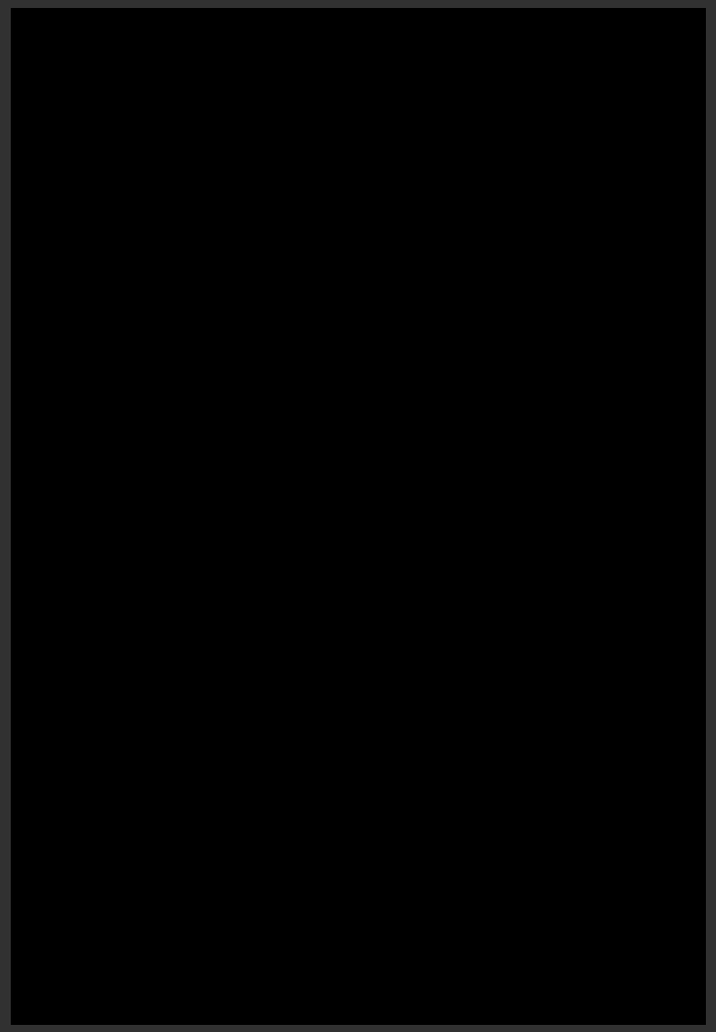
- This is also true for any function parameter

Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory entering main | line 15

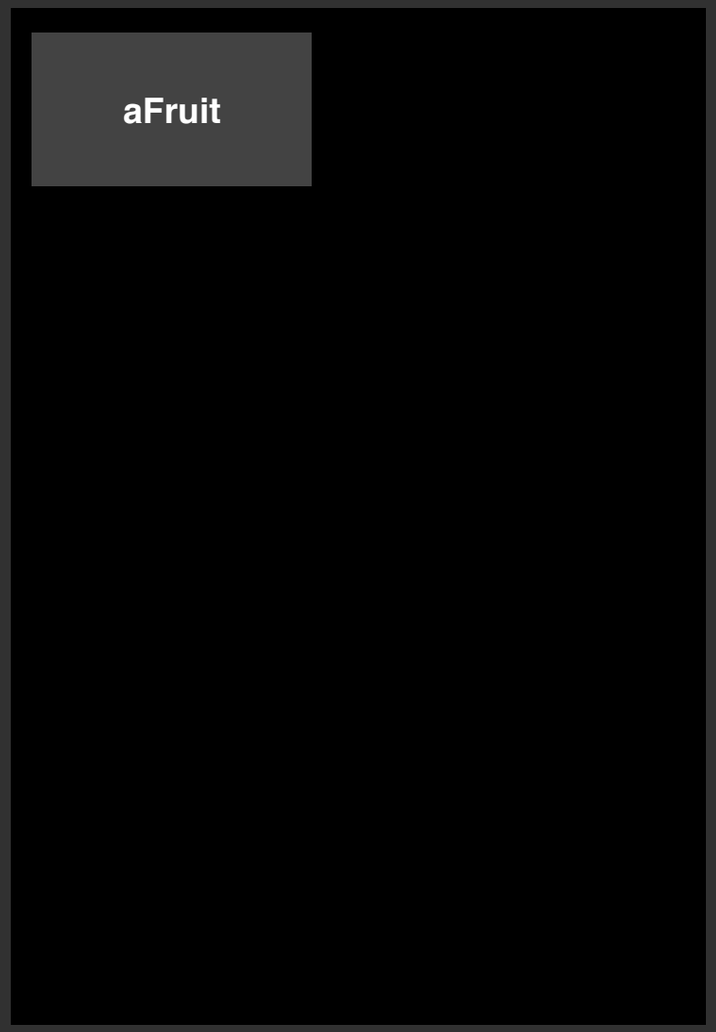


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory | line 16

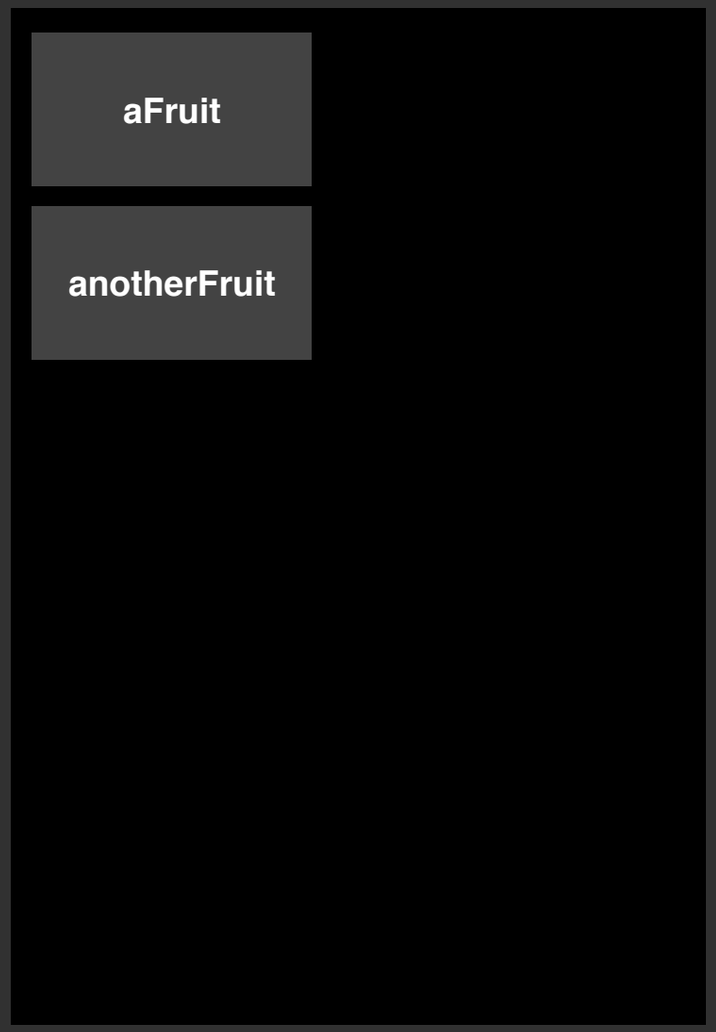


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory | line 17

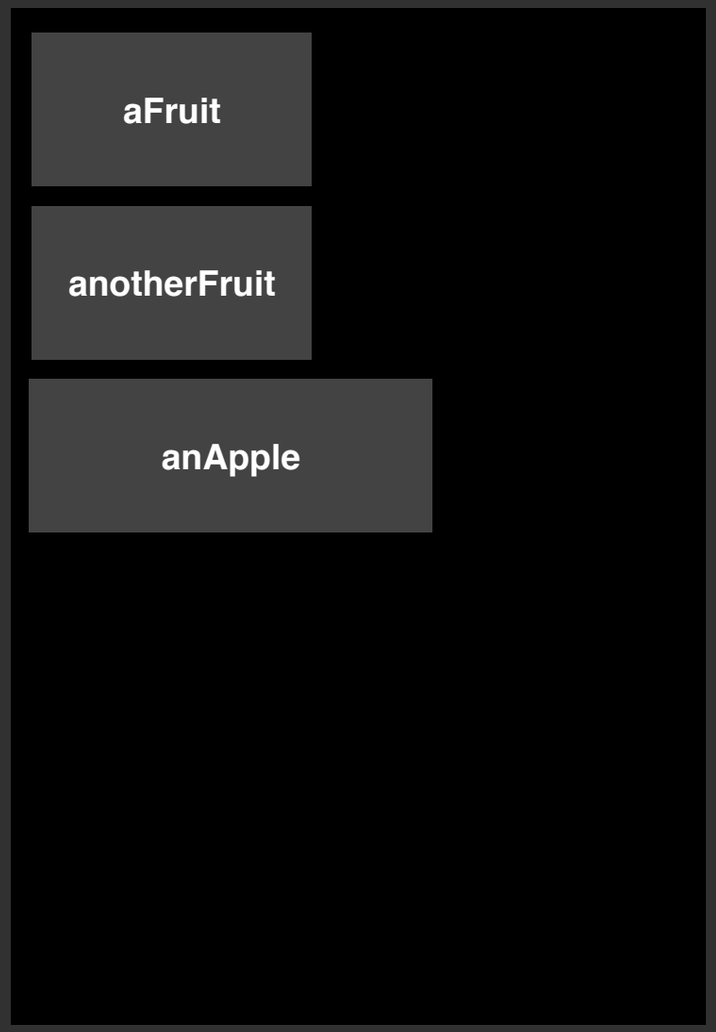


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory | line 18

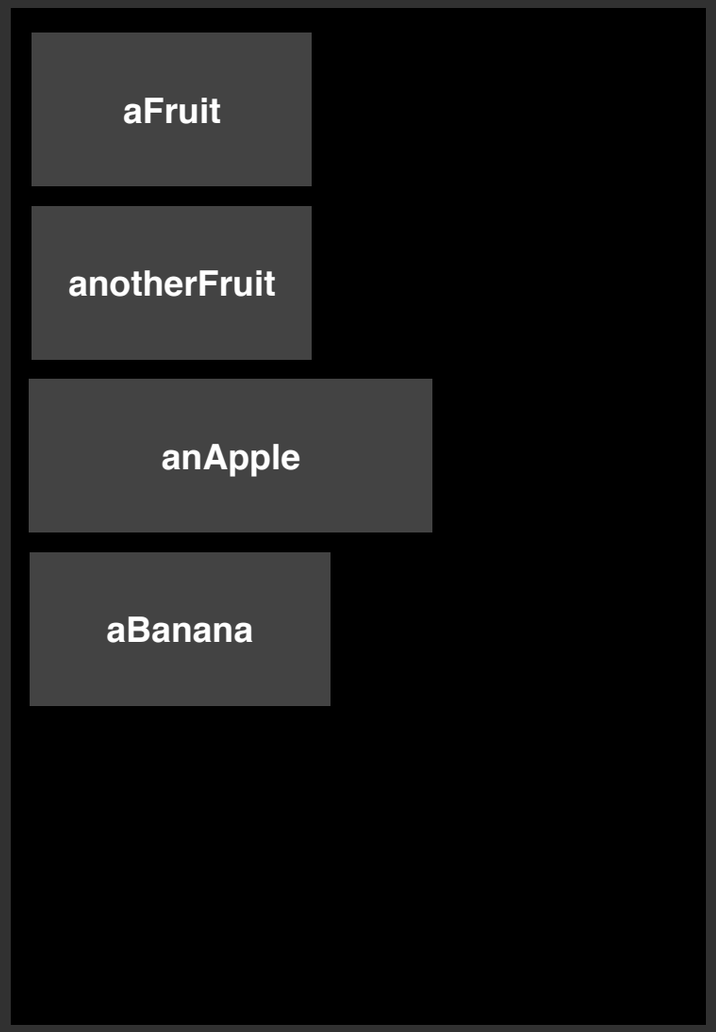


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory | line 19

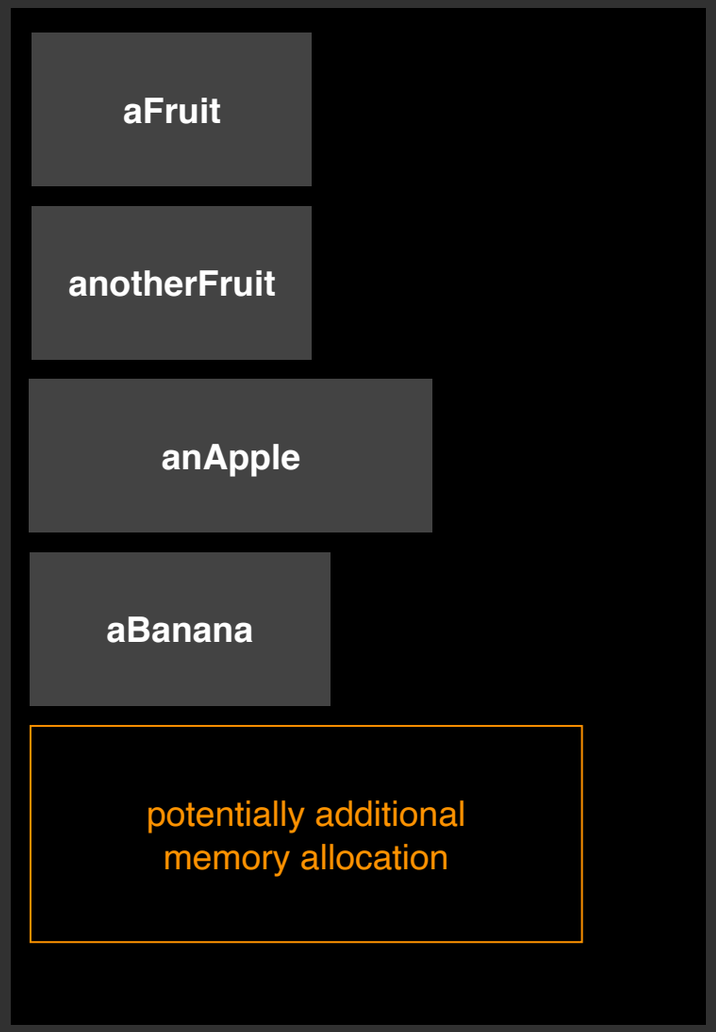


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory | lines 21 to 25

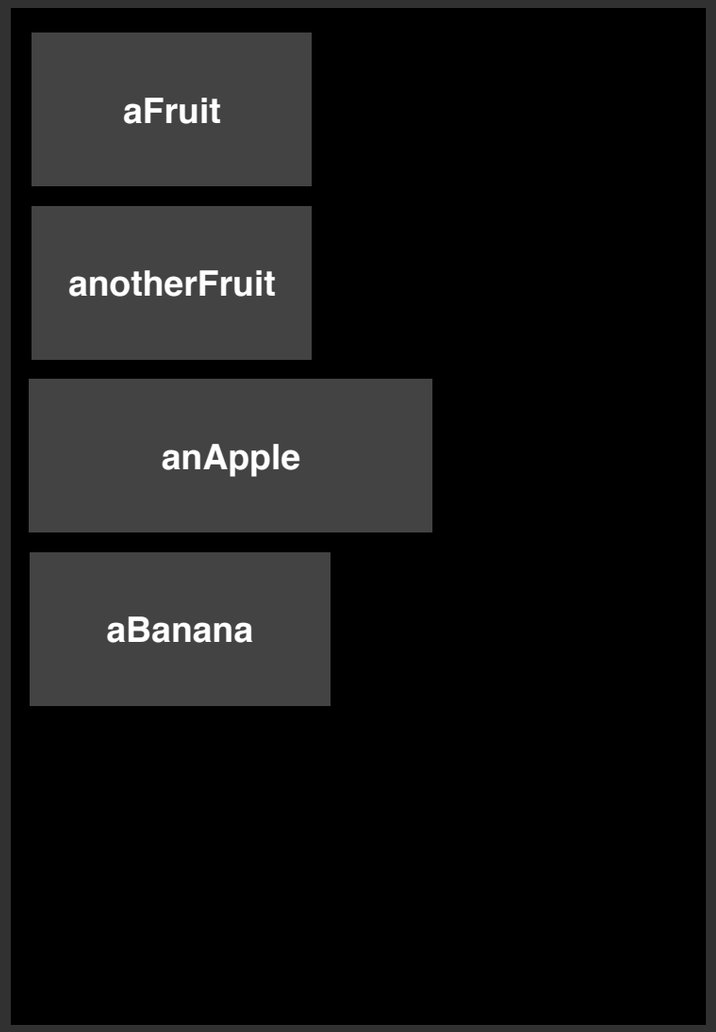


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory exiting main | line 28

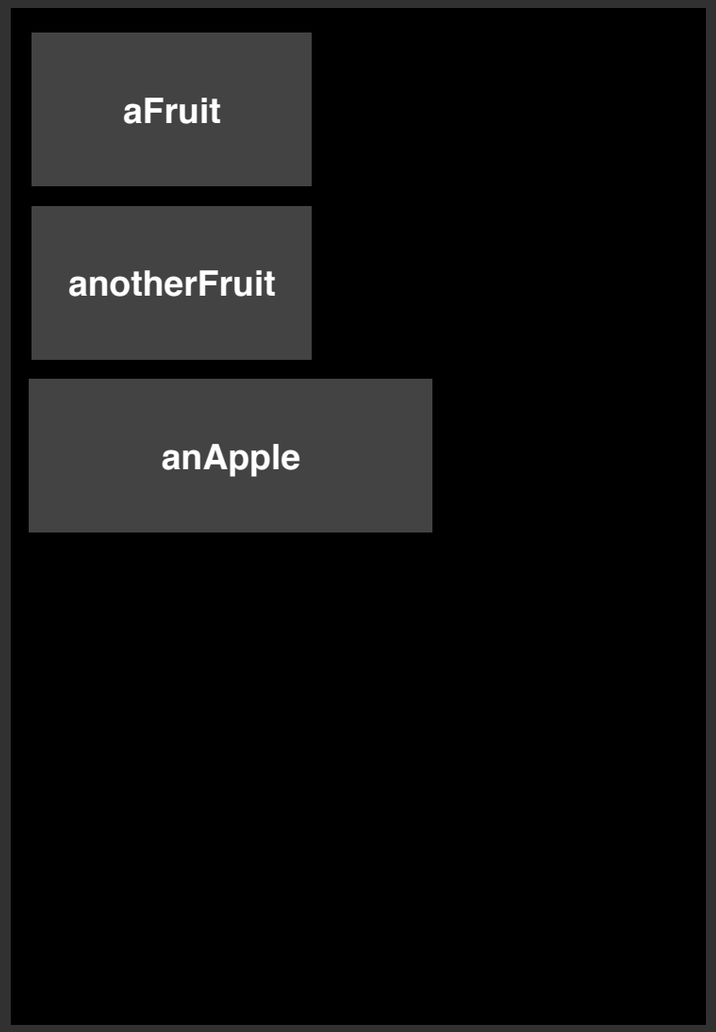


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory exiting main | line 28

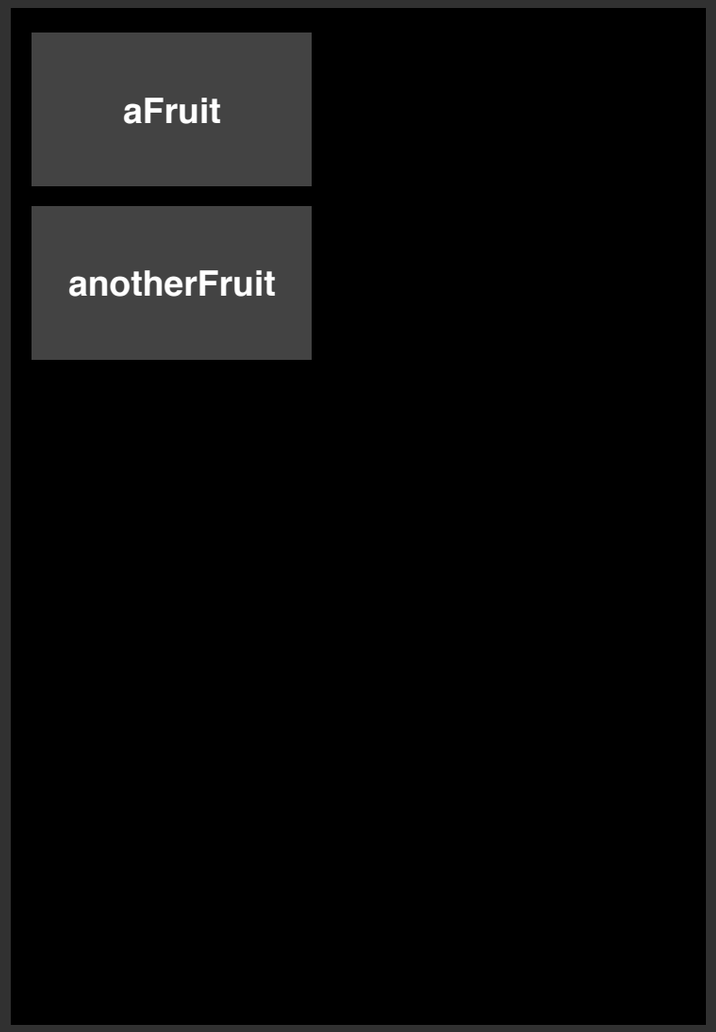


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory exiting main | line 28

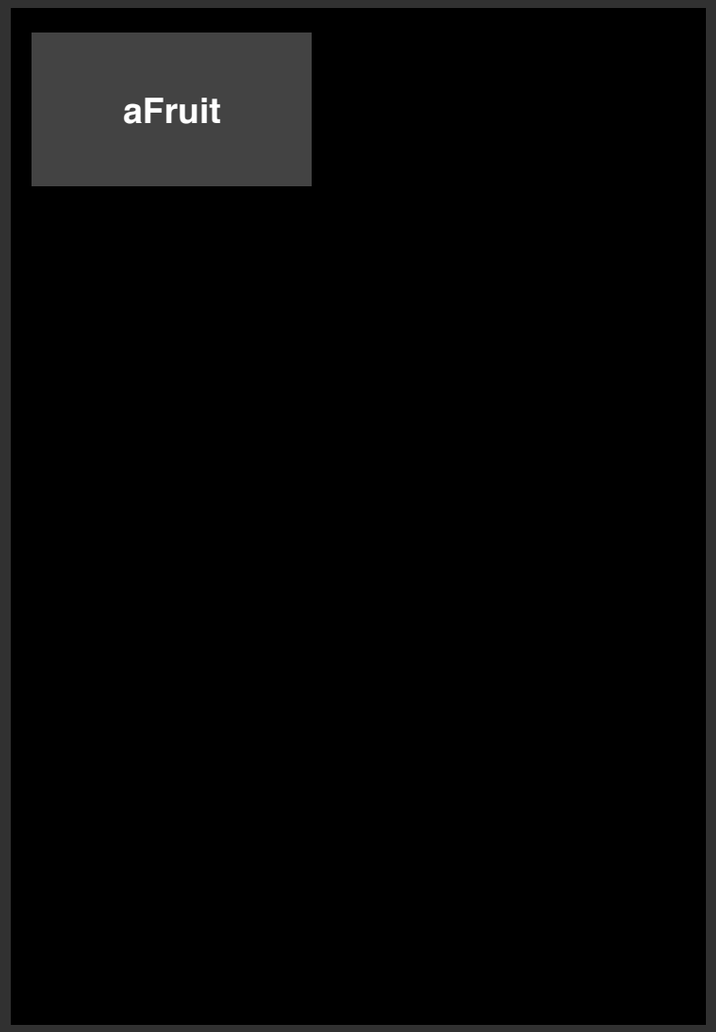


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory exiting main | line 28

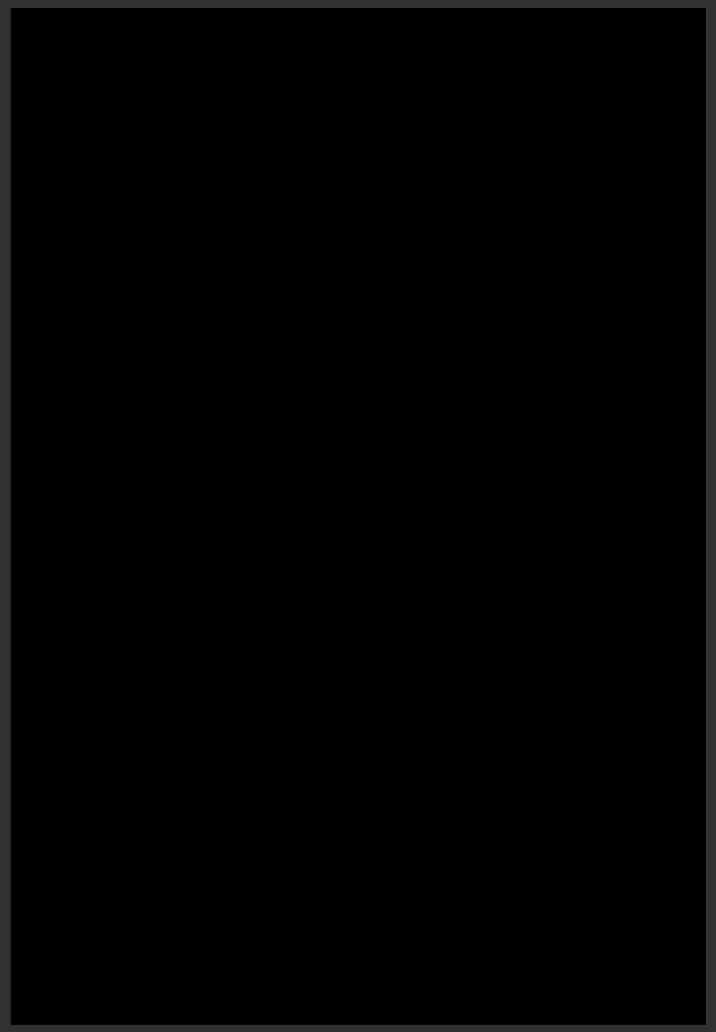


Local Variable’s Lifetime

source code instructions

```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
29
```

computer memory exiting main | line 29

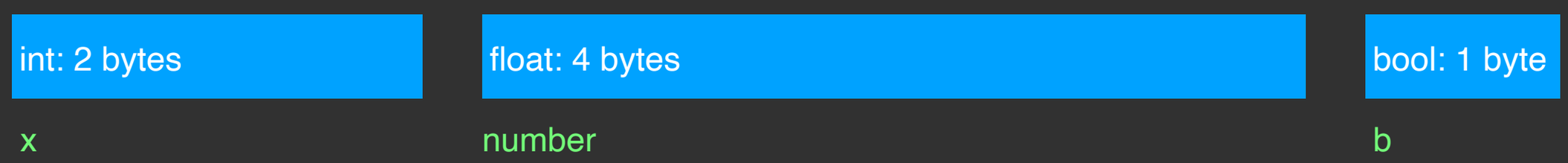


Memory Allocation & Access

- When a variable is declared, the required physical memory is allocated to store a value of the associated data type

- The necessary memory size depends on the type

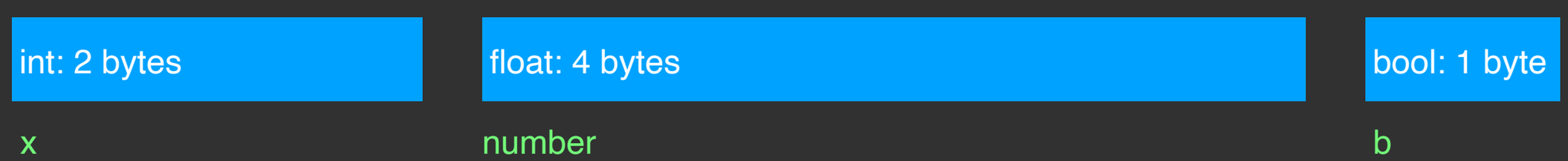
```
12
13 int x;
14 float number;
15 bool b;
16
```



Memory Allocation & Access

- In memory, the memory blocks are identified by an address
- In the code, this address can be obtained by using the address-of-operator &

```
17
18 cout << "Address of x is " << &x << endl;          // 0x7fff5e92499c
19 cout << "Address of number is " << &number << endl; // 0x7fff5e924998
20 cout << "Address of b is " << &b << endl;          // 0x7fff5e924997
21
```



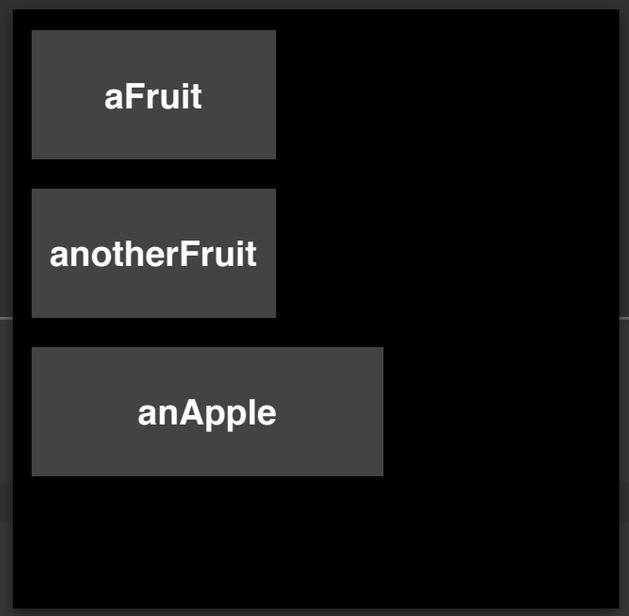
Automatic Memory Allocation

- Automatic memory allocation is an automated process of allocating & freeing memory for variables & data members
- Automatic memory allocation happens for local variables, class member variables and function parameters

Automatic Memory Allocation

- Memory is allocated automatically when the relevant block where the variable is defined is reached

- Memory is released automatically when the relevant block where the variable is defined is exited




```
9  #include "fruit.h"
10 #include "apple.h"
11 #include "banana.h"
12
13 |
14 int main()
15 {
16     fruit aFruit;
17     fruit anotherFruit("anotherFruit");
18     apple anApple{apple::appleType::BRAEBURN};
19     banana aBanana;
20
21     aFruit.printName();
22     anotherFruit.printName();
23
24     anApple.printName();    // the derived classes inherit the
25     aBanana.printName();    // base class function printName()
26
27     return 0;
28 }
```

Automatic Memory Allocation

- The memory area used by the compiler for automatic memory allocation is referred to as “The Stack”

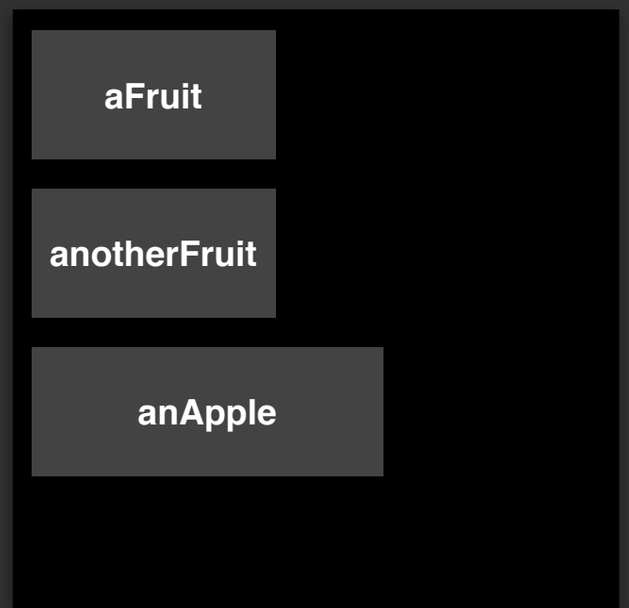


Image credit: https://blog.hypem.com/2014/06/hype-machine-stack/

Take Away

- Any variable that is used in a computer program requires the compiler to allocate space in the physical computer memory to store the variable values
- To do so efficiently, C++ compiler's require to know the variable data types (and values) at compile time
- The memory blocks are identified by an address which can be retrieved in the code by using the address-of-operator "&"
- In the case of local variables, function parameters, and class members, the allocation and de-allocation of memory is done automatically