

Variables

Last updated 11/23/20

Variables

- These slides introduce variables
- Upon completion: You should be able to interpret and code using variables

Variables

- Variable
 - Symbolic representation for a value - name
 - Stored in memory (data)
 - Can be modified during execution
- Since it requires space in memory it must have a type to tell the compiler how much space to reserve
- Allowed characters: **letters**, **numbers**, **_**
- Cannot begin with a number

Variables

- Variable Declaration
 - Specify the type and name for a variable
 - Must be declared before it can be used

```
int foo;  
float rate;  
char initial1;
```

```
int var1, this, is, not, a, good, practice;
```

```
int AccountBalance;  
int annual_interest_rate;
```

**** Note: name length has no impact on compiled program size
focus on readable code**

Variables

- Variable Initialization
 - Variables are not initialized just by declaring them
 - They do not automatically have a value of 0
 - They may well have garbage values

Nothing stops you from using an un-initialized variable

```
int foo = 23;
```

```
int foo, boo = 23;
```

```
int count;  
count = 0;
```

```
int foo = 23, boo = 23;
```

```
char fx = 'A';
```

```
float pie = 3.14159;
```

Variables

- Variables and Memory

```
int foo;  
char initial1;  
float rate;  
char initial 2;  
int boo;
```

reserved for foo
has garbage in it

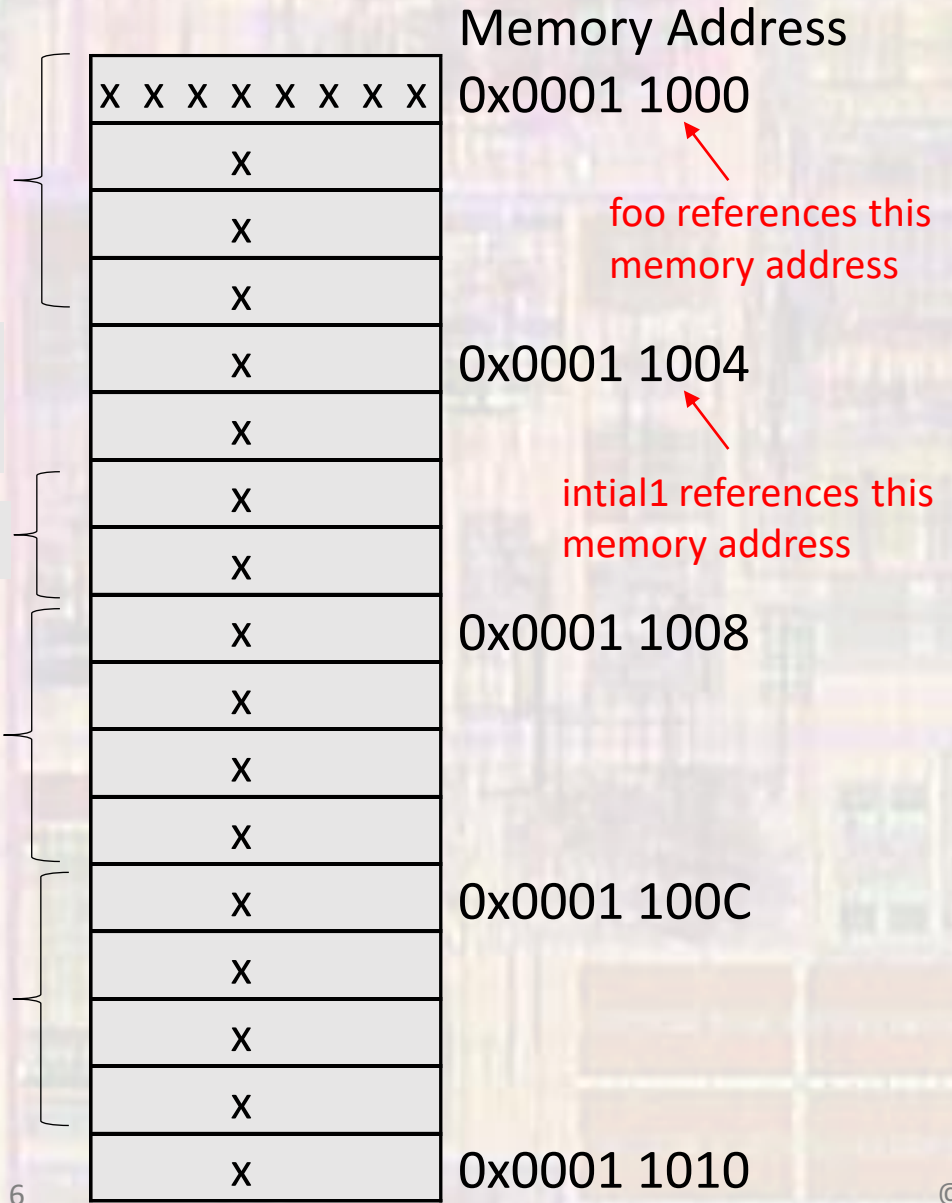
initial 1

initial 2

unused - alignment

rate

boo



Variables

- Variables and Memory

```
int foo;
char initial1;
float rate;
char initial 2;
int boo;
initial1 = 't';
rate = 2.5;
boo = 255;
```

reserved for foo
has garbage in it

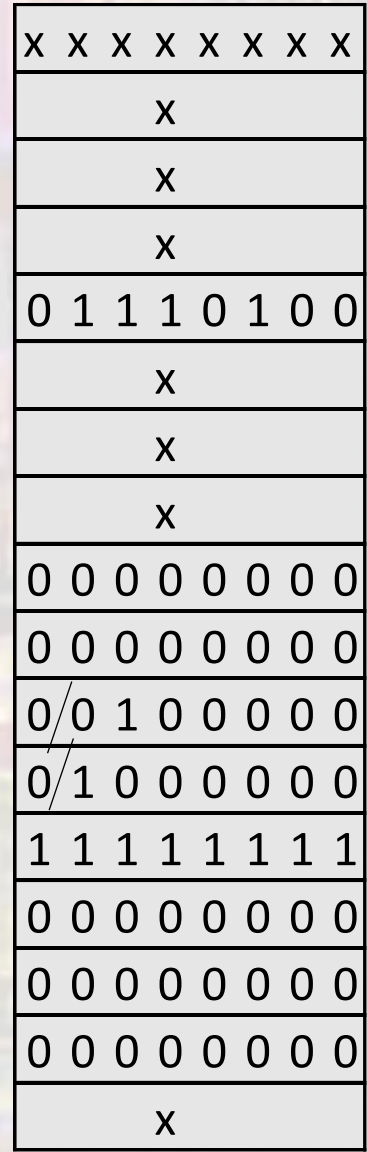
initial 1 – 't' – hex 0x74

initial 2 - garbage

unused - alignment

2.5 → 10.1 binary →
1.01 x 2¹ → 0 sign
01 mantissa
1000000 exponent

boo



Memory Address

0x0001 1000

foo references this
memory address

0x0001 1004

initial1 references this
memory address

0x0001 1008

0x0001 100C

0x0001 1010