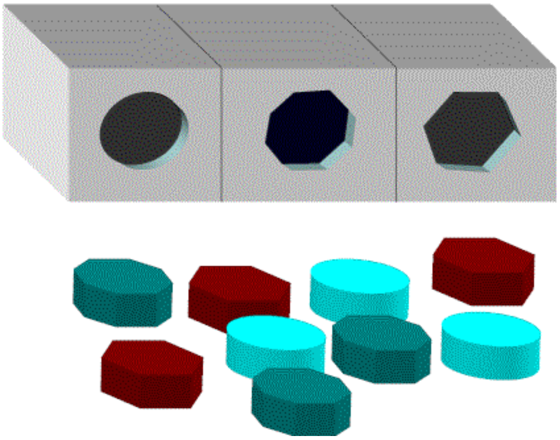
IS12 - Introduction to Programming

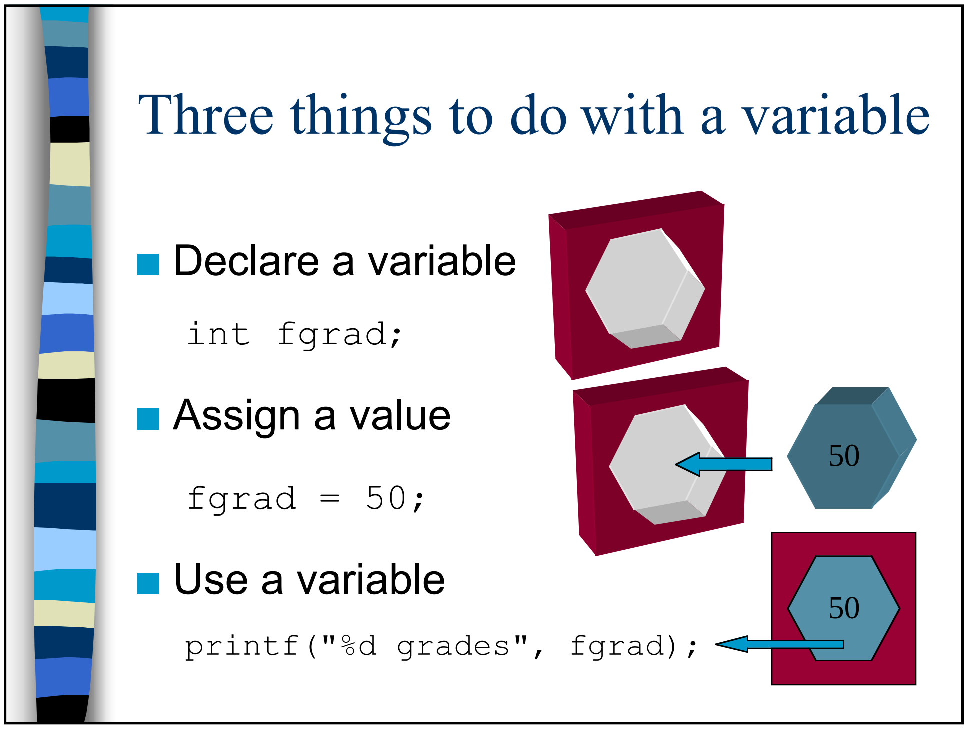
Lecture 9: Variables

Peter Brusilovsky

http://www2.sis.pitt.edu/~peterb/0012-051/

Data Types and Variables





Three things to do with a variable

- Declare a variable

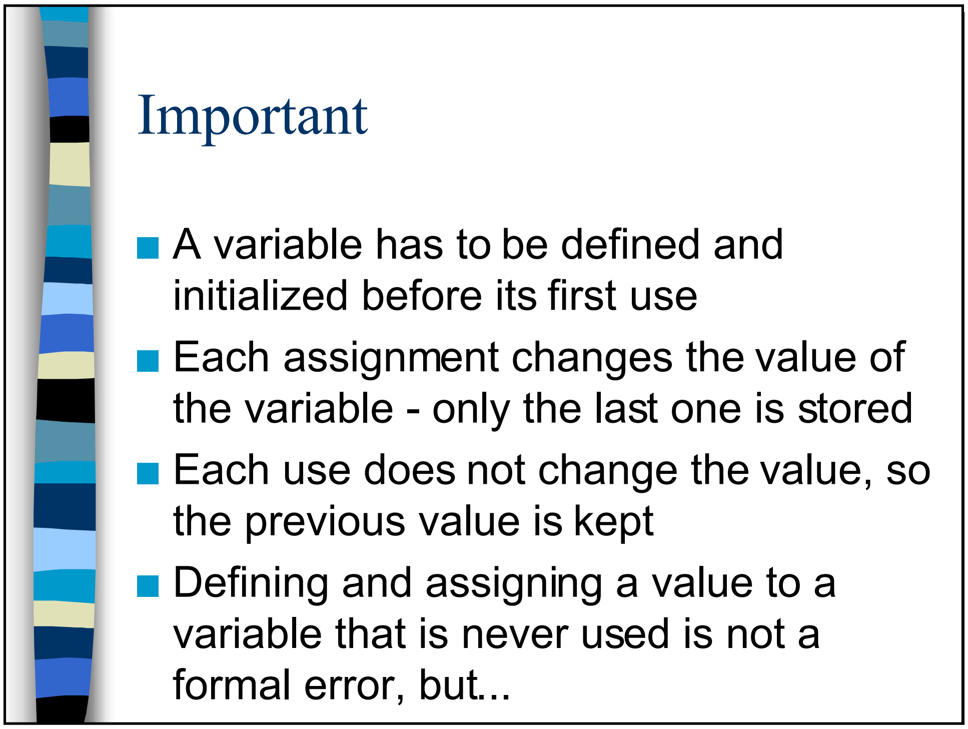
```
int fgrad;
```

- Assign a value

```
fgrad = 50;
```

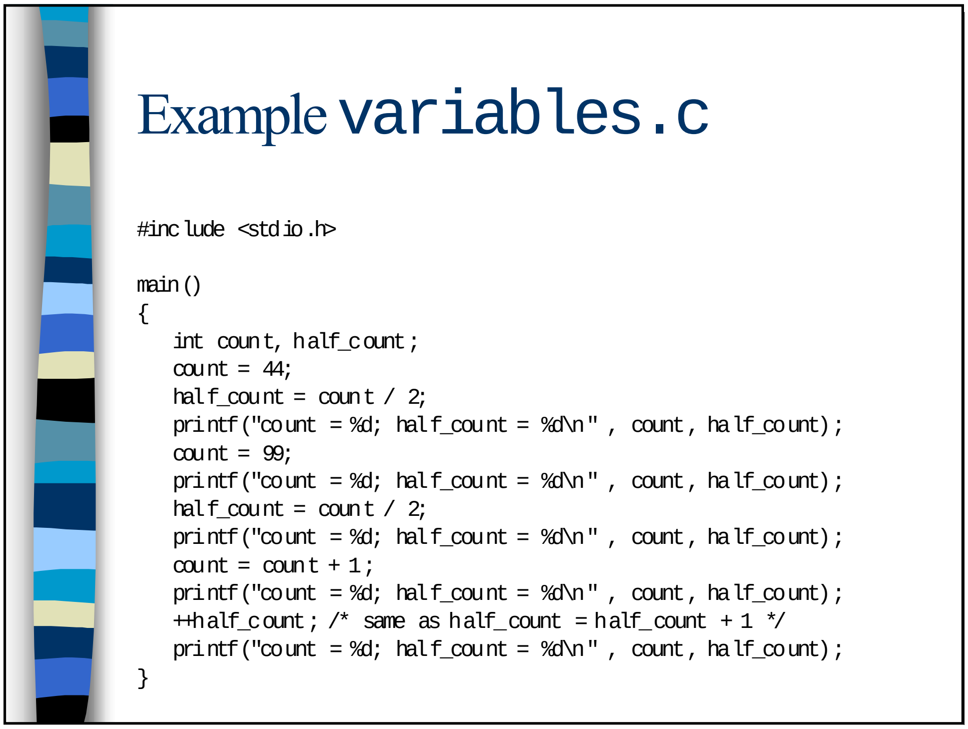
- Use a variable

```
printf("%d grades", fgrad);
```



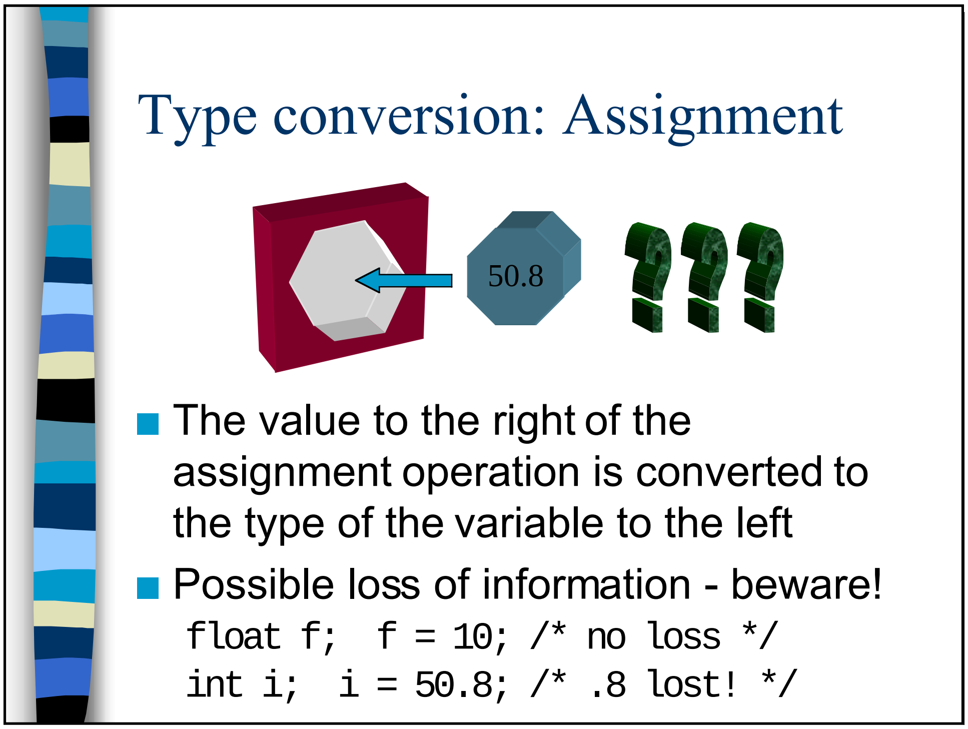
Important

- A variable has to be defined and initialized before its first use
- Each assignment changes the value of the variable - only the last one is stored
- Each use does not change the value, so the previous value is kept
- Defining and assigning a value to a variable that is never used is not a formal error, but...



Example `variables.c`

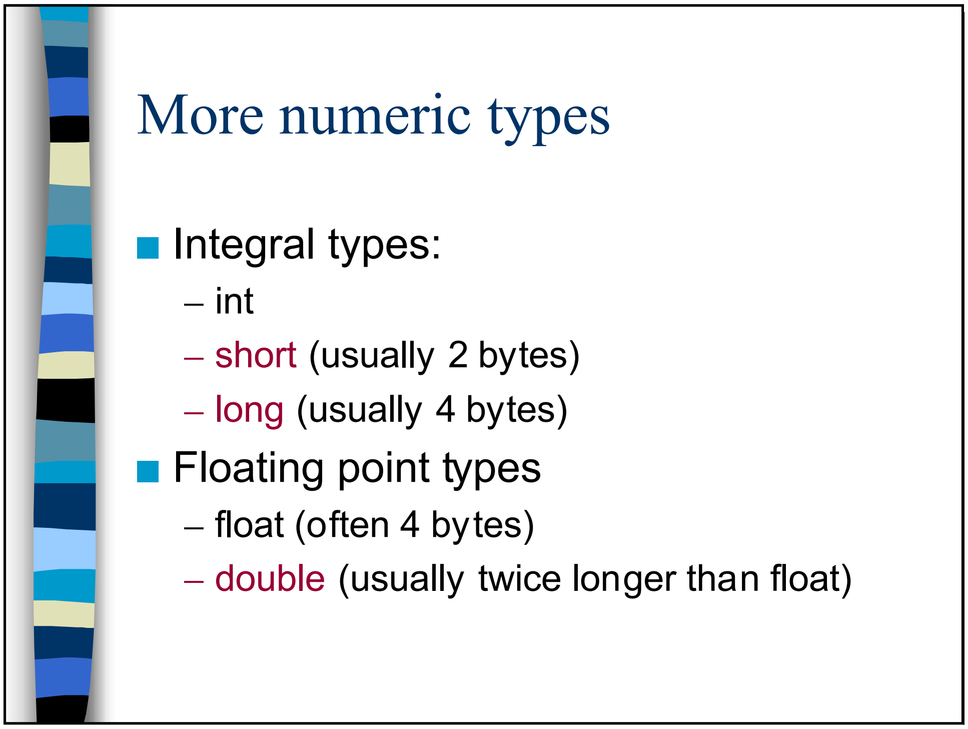
```
#include <stdio.h>

main()
{
   int count, half_count;
   count = 44;
   half_count = count / 2;
   printf("count = %d; half_count = %d\n", count, half_count);
   count = 99;
   printf("count = %d; half_count = %d\n", count, half_count);
   half_count = count / 2;
   printf("count = %d; half_count = %d\n", count, half_count);
   count = count + 1;
   printf("count = %d; half_count = %d\n", count, half_count);
   ++half_count; /* same as half_count = half_count + 1 */
   printf("count = %d; half_count = %d\n", count, half_count);
}
```



Type conversion: Assignment

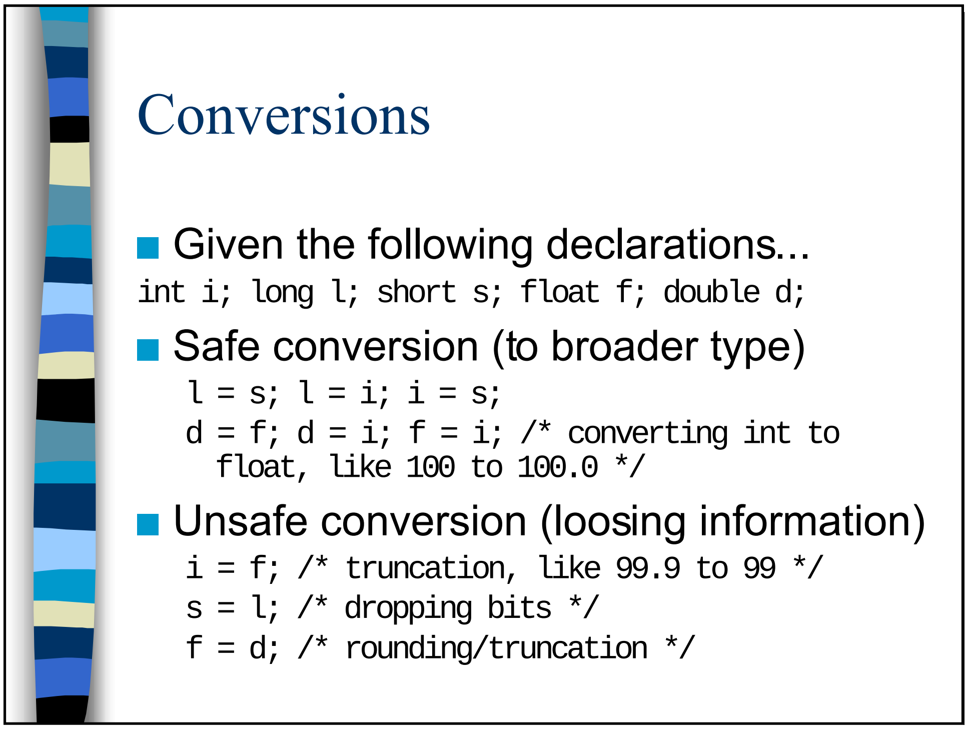
- The value to the right of the assignment operation is converted to the type of the variable to the left
- Possible loss of information - beware!

```
float f;  f = 10; /* no loss */
int i;  i = 50.8; /* .8 lost! */
```



More numeric types

- Integral types:
  - int
  - short (usually 2 bytes)
  - long (usually 4 bytes)
- Floating point types
  - float (often 4 bytes)
  - double (usually twice longer than float)



Conversions

- Given the following declarations...

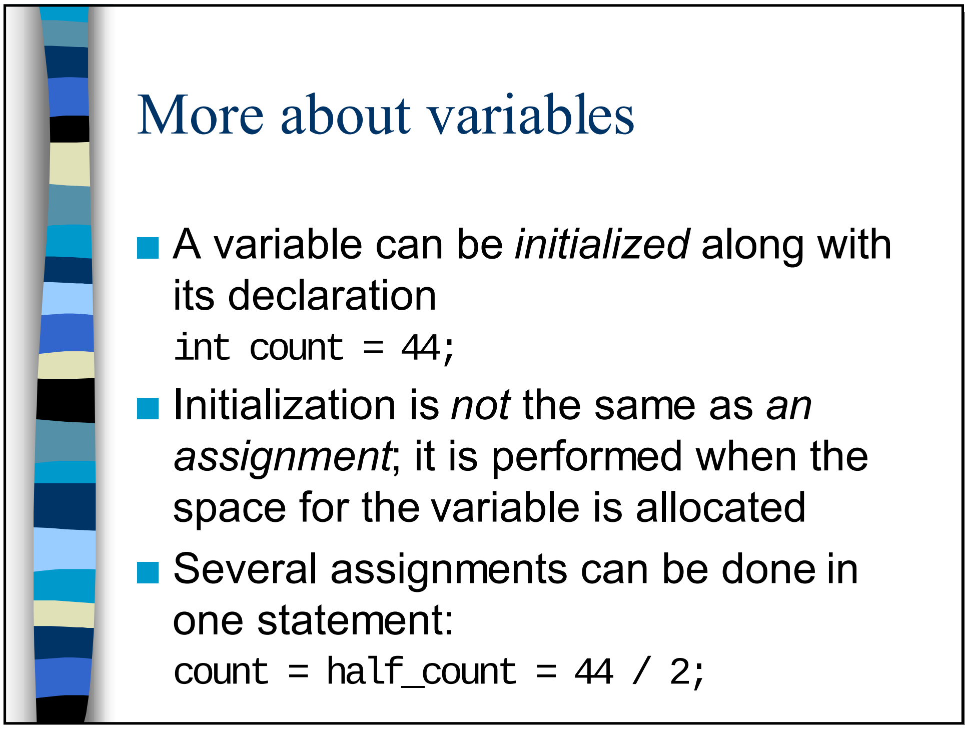
```
int i; long l; short s; float f; double d;
```

- Safe conversion (to broader type)

```
l = s; l = i; i = s;
d = f; d = i; f = i; /* converting int to
  float, like 100 to 100.0 */
```

- Unsafe conversion (loosing information)

```
i = f; /* truncation, like 99.9 to 99 */
s = l; /* dropping bits */
f = d; /* rounding/truncation */
```



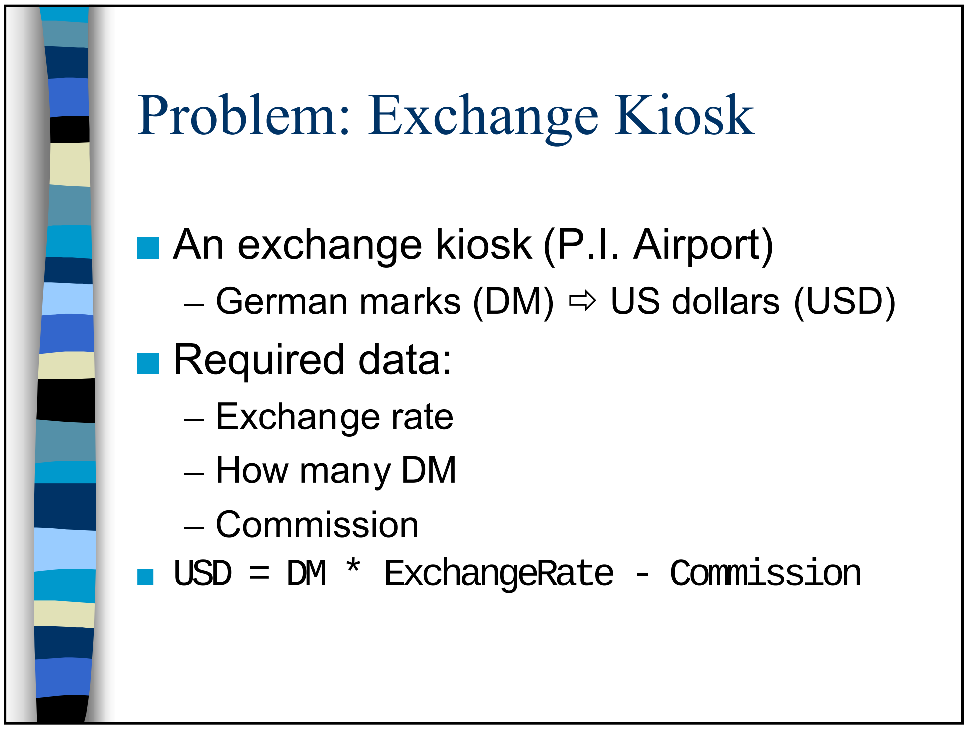
More about variables

- A variable can be *initialized* along with its declaration

```
int count = 44;
```

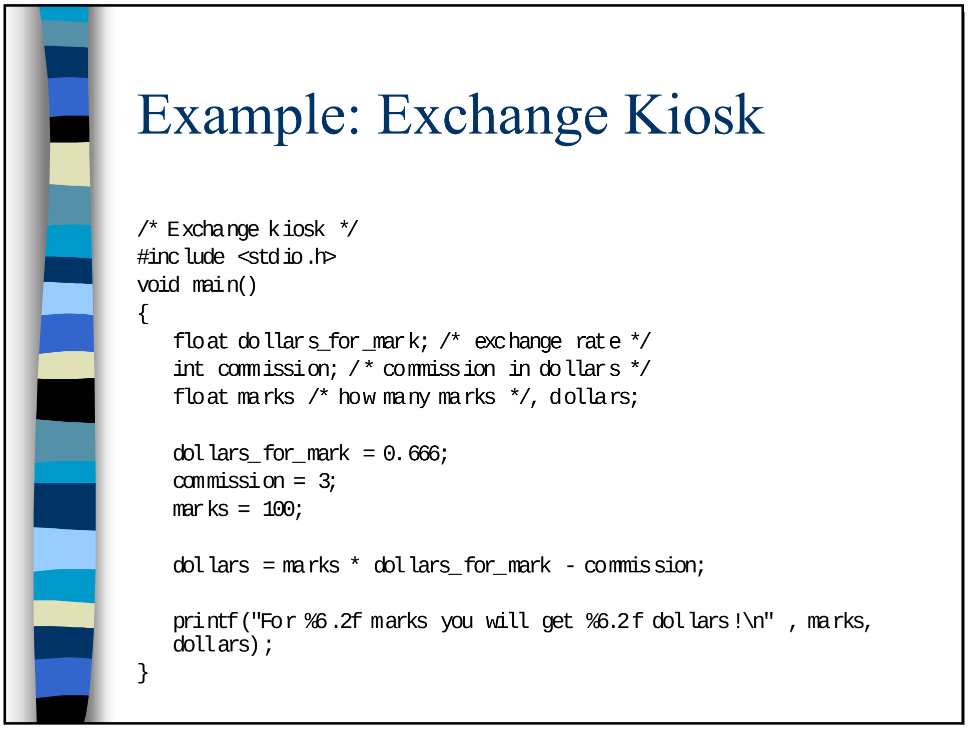
- Initialization is *not* the same as *an assignment*; it is performed when the space for the variable is allocated
- Several assignments can be done in one statement:

```
count = half_count = 44 / 2;
```



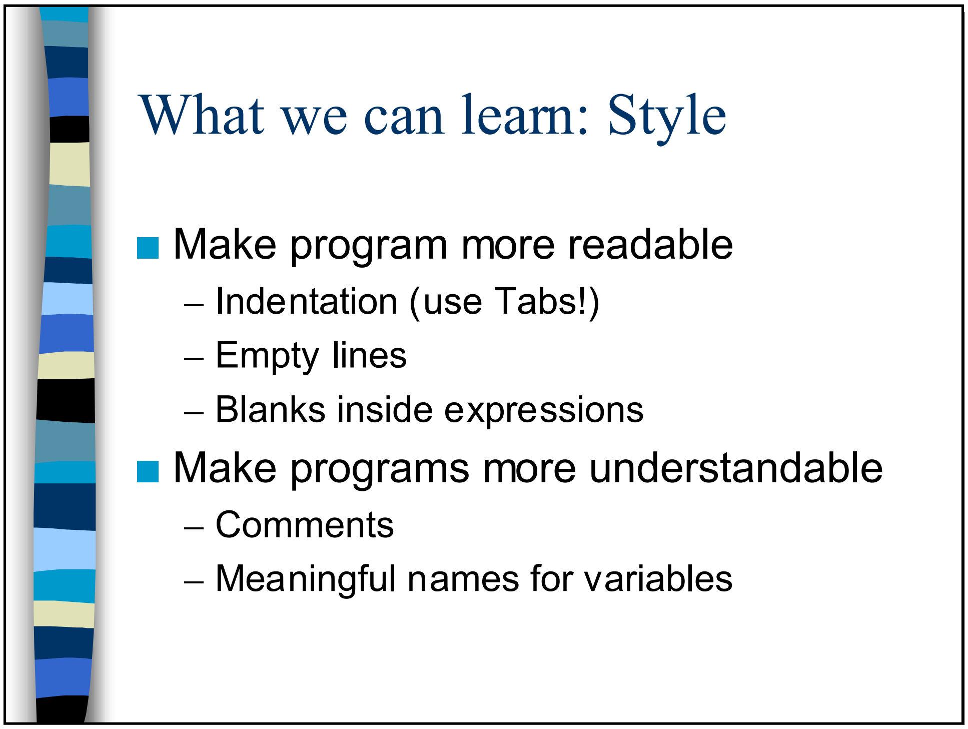
Problem: Exchange Kiosk

- An exchange kiosk (P.I. Airport)
  - German marks (DM) ⇨ US dollars (USD)
- Required data:
  - Exchange rate
  - How many DM
  - Commission
- `USD = DM * ExchangeRate - Commission`



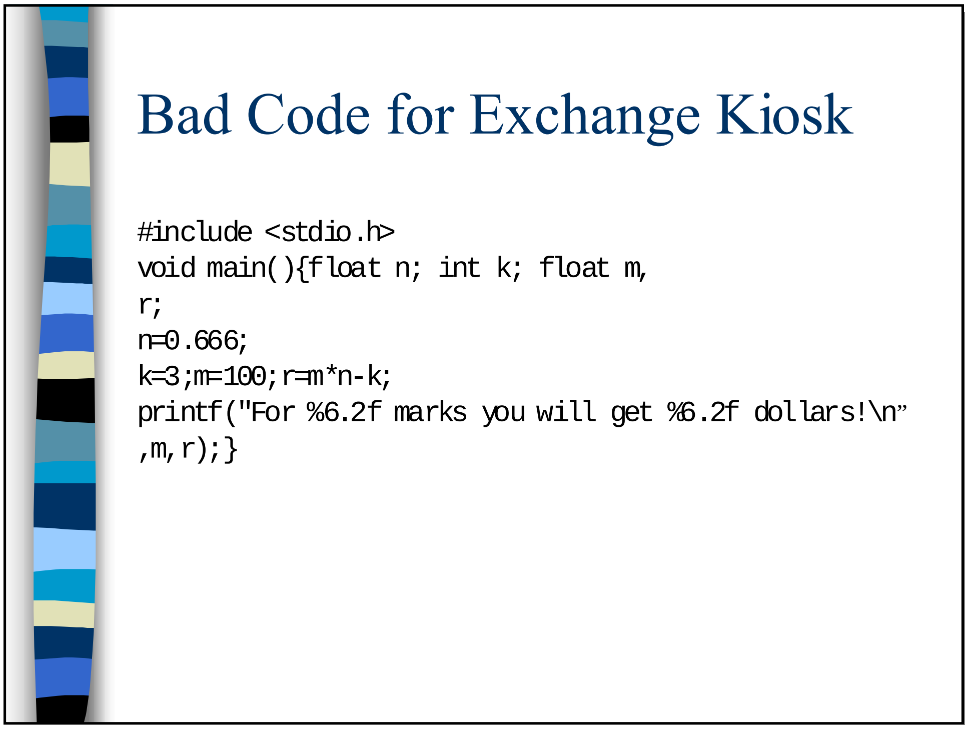
Example: Exchange Kiosk

```
/* Exchange kiosk */
#include <stdio.h>
void main()
{
   float dollars_for_mark; /* exchange rate */
   int commission; /* commission in dollars */
   float marks /* how many marks */, dollars;

   dollars_for_mark = 0.666;
   commission = 3;
   marks = 100;

   dollars = marks * dollars_for_mark - commission;

   printf("For %6.2f marks you will get %6.2f dollars!\n" , marks,
   dollars);
}
```



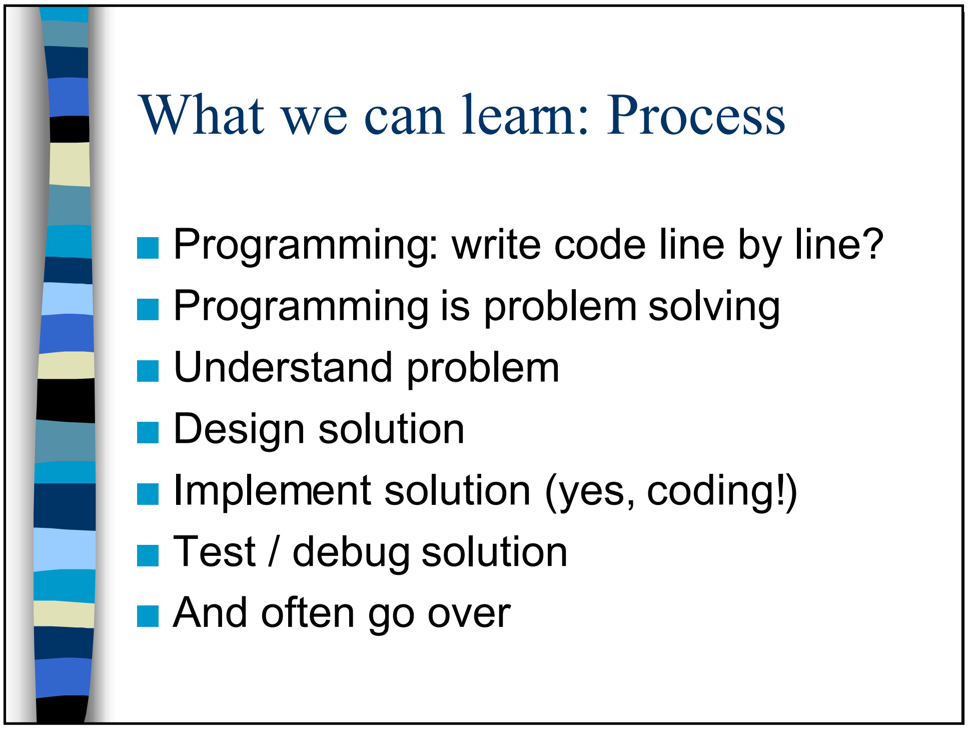
What we can learn: Style

- Make program more readable
  - Indentation (use Tabs!)
  - Empty lines
  - Blanks inside expressions
- Make programs more understandable
  - Comments
  - Meaningful names for variables



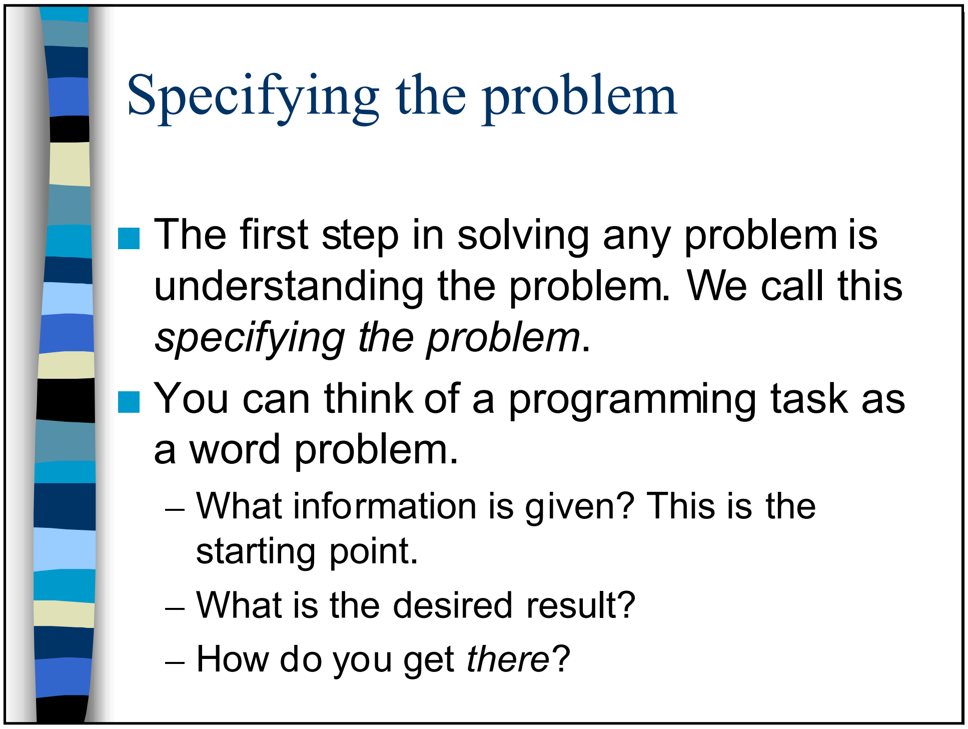
Bad Code for Exchange Kiosk

```
#include <stdio.h>
void main(){float n; int k; float m,
r;
n=0.666;
k=3;m=100;r=m*n-k;
printf("For %6.2f marks you will get %6.2f dollars!\n”
,m,r);}
```



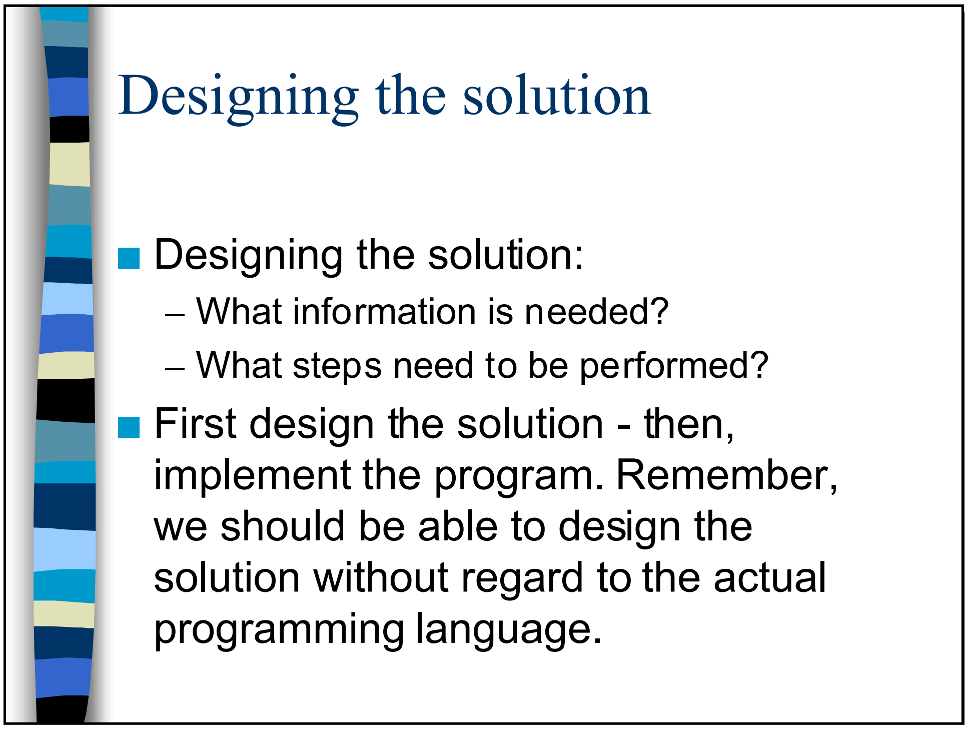
What we can learn: Process

- Programming: write code line by line?
- Programming is problem solving
- Understand problem
- Design solution
- Implement solution (yes, coding!)
- Test / debug solution
- And often go over



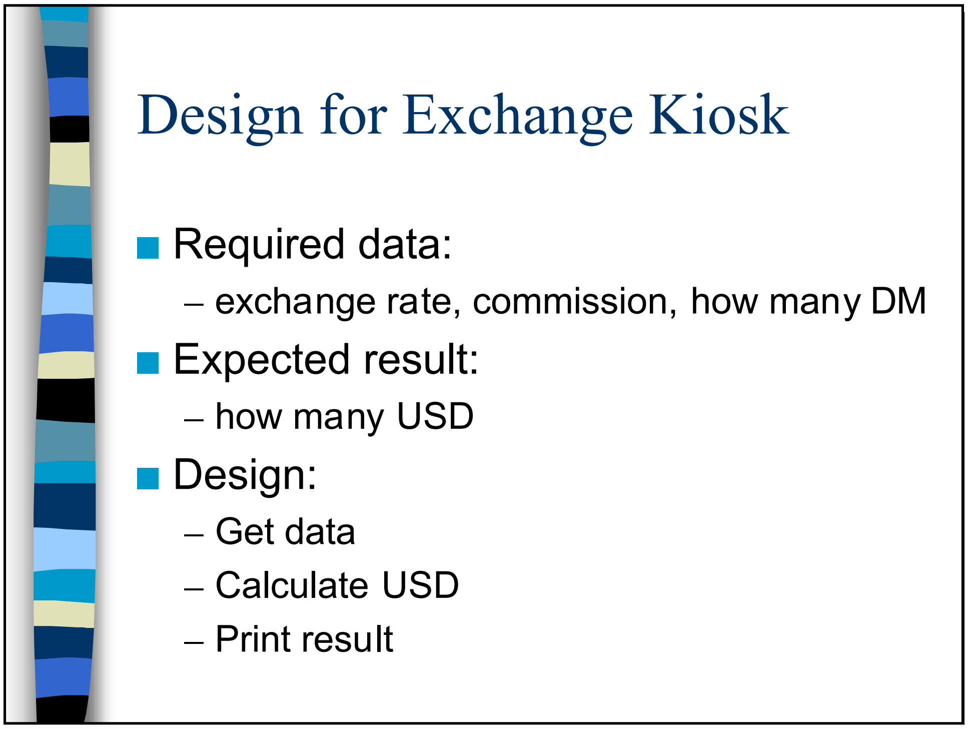
Specifying the problem

- The first step in solving any problem is understanding the problem. We call this *specifying the problem*.
- You can think of a programming task as a word problem.
  - What information is given? This is the starting point.
  - What is the desired result?
  - How do you get *there*?



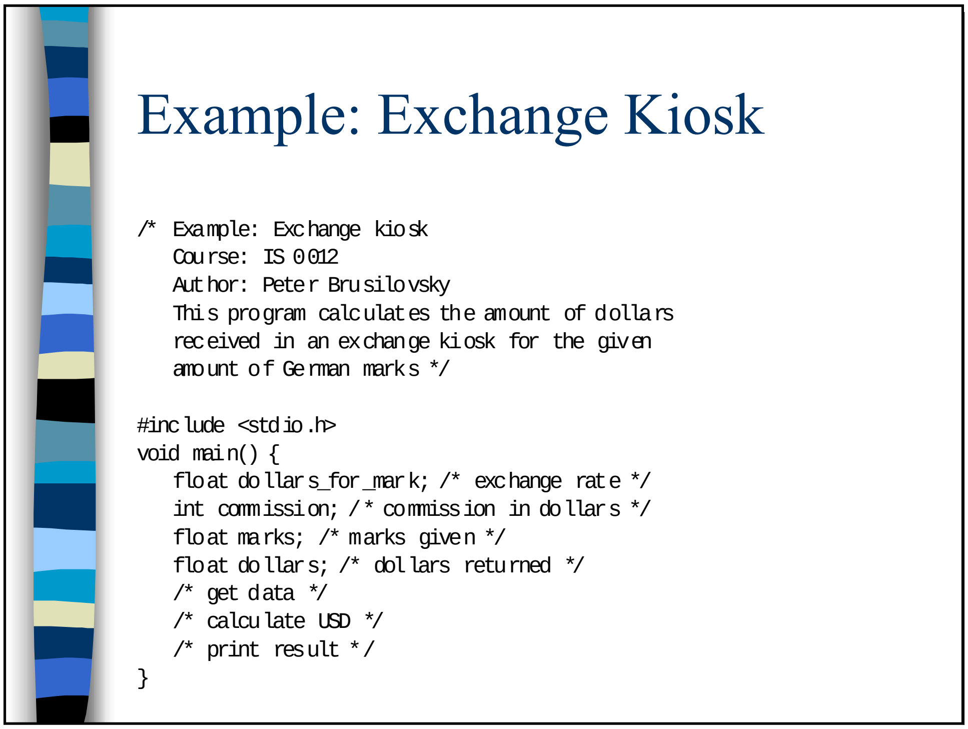
Designing the solution

- Designing the solution:
  - What information is needed?
  - What steps need to be performed?
- First design the solution - then, implement the program. Remember, we should be able to design the solution without regard to the actual programming language.



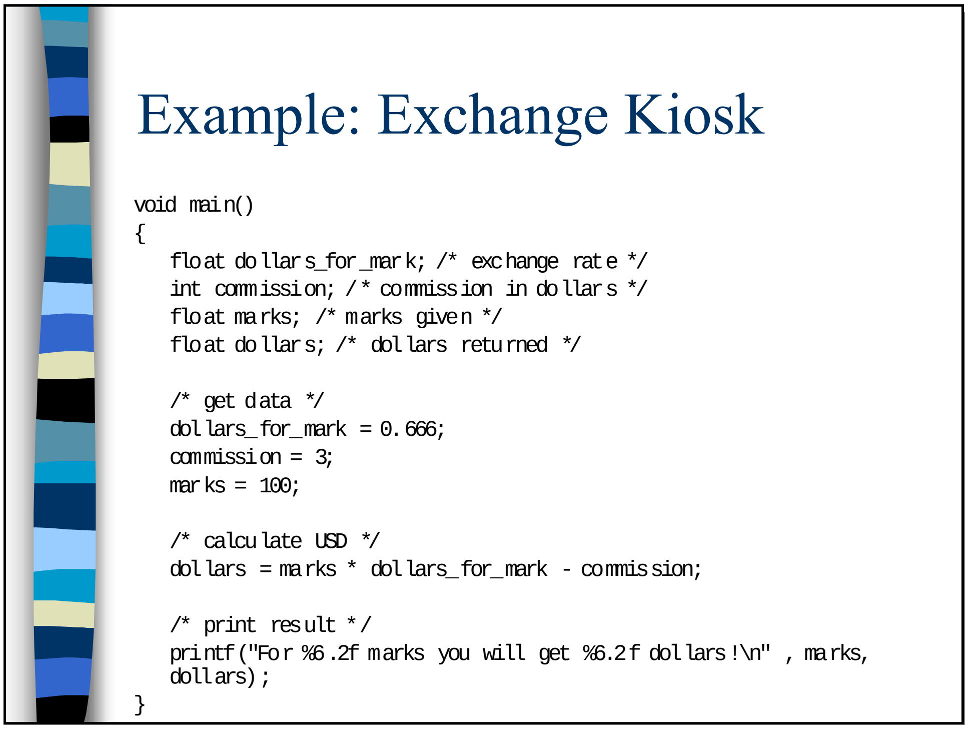
Design for Exchange Kiosk

- Required data:
  - exchange rate, commission, how many DM
- Expected result:
  - how many USD
- Design:
  - Get data
  - Calculate USD
  - Print result



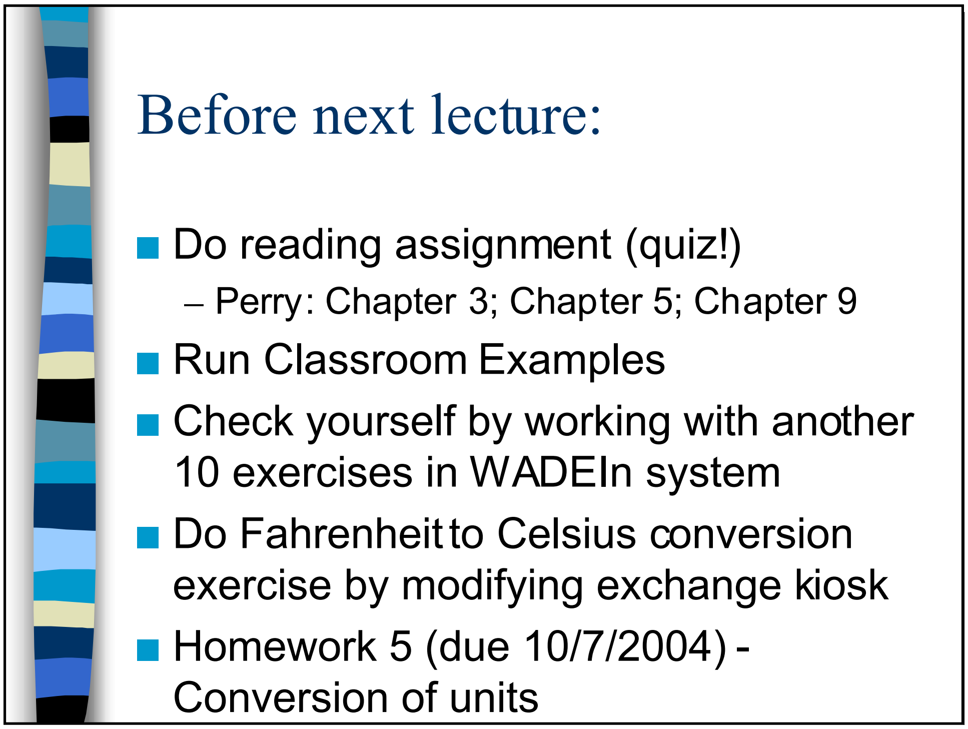
Example: Exchange Kiosk

```
/* Example: Exchange kiosk
   Course: IS 0012
   Author: Peter Brusilovsky
   This program calculates the amount of dollars
   received in an exchange kiosk for the given
   amount of German marks */

#include <stdio.h>
void main() {
   float dollars_for_mark; /* exchange rate */
   int commission; /* commission in dollars */
   float marks; /* marks given */
   float dollars; /* dollars returned */
   /* get data */
   /* calculate USD */
   /* print result */
}
```



Example: Exchange Kiosk

```
void main()
{
   float dollars_for_mark; /* exchange rate */
   int commission; /* commission in dollars */
   float marks; /* marks given */
   float dollars; /* dollars returned */

   /* get data */
   dollars_for_mark = 0.666;
   commission = 3;
   marks = 100;

   /* calculate USD */
   dollars = marks * dollars_for_mark - commission;

   /* print result */
   printf("For %6.2f marks you will get %6.2f dollars!\n" , marks,
   dollars);
}
```



Before next lecture:

- Do reading assignment (quiz!)
  - Perry: Chapter 3; Chapter 5; Chapter 9
- Run Classroom Examples
- Check yourself by working with another 10 exercises in WADEIn system
- Do Fahrenheit to Celsius conversion exercise by modifying exchange kiosk
- Homework 5 (due 10/7/2004) - Conversion of units