IS12 - Introduction to Programming

Lecture 9: Variables

Peter Brusilovsky
http://www2.sis.pitt.edu/~peterb/0012-072/

Data Types and Variables
Three things to do with a variable

- Declare a variable
  ```
  int age;
  ```

- Assign a value
  ```
  age = 50;
  ```

- Use a variable
  ```
  printf("%d years old", age);
  ```

Example `variables.c`

```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);
}```
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...

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`
  - `long int` (holds more digits)
- Floating point types
  - `float`
  - `double` (holds more digits)

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
  ```c
  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:
  ```c
  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

```c
/* 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 %.2f marks you will get %.2f dollars!\n", marks, dollars);
}
```

Functions in C

- Functions - a way to implement a part of the whole job and forget about details
- Functions are called using names and parameters; they return values
- C makes function calls efficient
- There are standard functions (printf, scanf) and user-defined functions
Karel Commands vs. C Functions

- Same control mechanism
  - calling a separate program fragment
  - same command/function can be called from several places
  - after command/function is executed, the control returns to the line after call
- Karel commands had no parameters - can do exactly the same thing each call
- C functions can return a value and can have parameters (we will learn about parameters later)

Functions and Global Variables

- The need for shared data
  - Karel world was accessible by all commands, so each command can access and change the world
  - How functions can share data in C?
- Global Variables
  - C supports global variables
  - Global variables should be defined outside of all functions - they are top-level objects
  - Global variables are accessible by all functions
Example: Exchange Kiosk

```c
/* Exchange kiosk */
#include <stdio.h>
double marks /* how many marks */;

void main() {
    double dollars;
    marks = 100.0;
    dollars = convert();
    printf("For %6.2f marks you will get %6.2f dollars!\n", marks, dollars);
    marks = 300.0;
    printf("For %6.2f marks you will get %6.2f dollars!\n", marks, convert());
}

double convert() {
    double dollars_for_mark; /* exchange rate */
    int commission; /* commission in dollars */
    dollars_for_mark = 0.666;
    commission = 3;
    return (marks * dollars_for_mark - commission);
}
```

Exchange: Control flow

```
main() {
    exchange();
    double convert() {
        return (...);
    }

    exchange();
    double convert() {
        return (...);
    }
}
```
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
- Work with QuizGuide
- Do Fahrenheit to Celsius conversion exercise by modifying exchange kiosk