Principles of Computer Science II

Nadeem Abdul Hamid

CSC121A - Spring 2005

Lecture Slides 9 - Lexical Elements, Operators, and the C System

Review

- Writing a simple C program with a main function
- Basic C types and values
- Variable declarations and assignment
- Input/output with scanf/printf

Lexical Elements

- (Programming) Languages have an alphabet and rules for putting together words and punctuation to form correct, or legal, programs
- These rules are the *syntax*
- Compiler goes through several stages
  - First preprocessor is invoked (as separate program, or integrated into compiler)
  - Compiler collects characters of the program to form *tokens* (words/basic vocabulary)
  - Compiler checks that the tokens are arranged into legal statements according to the syntax of the language (*parsing*)
  - Converts the program into object code and final executable code to run on a particular machine

Lexical Elements

- Characters recognized by C compiler
  - Uppercase/lowercase letters, digits, +-*/=(){}[]<>'"!@#$%&_|^-\.,;:?, whitespace (blank, newline, tab)
- Characters are collected into tokens (separated by whitespace)
  - Keywords, identifiers, constants, string constants, operators, punctuators

Sum Program

Compiler replaces comments with a single blank

```
/* Read in two integers and print their sum. */

#include <stdio.h>

int main(void)
{
   int   a, b, sum;

   printf("Input two integers:  ");
   scanf("%d%d", &a, &b);
   sum = a + b;
   printf("%d + %d = %d\n", a, b, sum);
   return 0;
}
```

Preprocessing directive: causes stdio.h to be included - contains function prototypes for printf *and* scanf

- Identifiers
  - main a b sum printf scanf
- Keywords
  - int return void
- Operators
  - ( ) + & =
- Punctuators
  - { } , ;
- Constants
  - 0
- String constants
  - "Input two integers: "
  - "%d%d"
  - "%d + %d = %d\n"

- Comments should be simultaneously written with program text
- Problems with inserting them later
  - Once program is written, tend to leave them or abbreviate them
  - Become inconsistent with the code instead of contributing to program clarity and correctness

Keywords

```
auto      do      goto      signed   unsigned
break     double  if        sizeof   void
case      else    int       static   volatile
char      enum    long      struct   while
const     extern  register  switch
continue  float   return    typedef
default   for     short     union
```

- Explicitly reserved words having a strict meaning in C
- Cannot be redefined or used in other contexts
- Some implementations provide additional keywords

Identifiers

- Token composed of letters, digits, _
- First character must be letter or underscore
- Upper and lower-case are distinct
- In ANSI C, only first 31 characters of identifiers are discriminated
- Avoid using identifiers starting with _
  - Used mostly by the system libraries

Constants (Literals)

- Integer
  - Can be written in decimal, hex, or octal
  - Decimal: 0 8 10 81
  - Octal: 000 010 012 0121
  - Hex: 0x0 0x8 0xA 0x51
- Character
  - 'a' '5' '\n' '\t'
- Floating-Point
  - 1.5 5. .7 0.75 1.2E5 1.25F 5.6L 7.8e-2L
- Enumeration (ch. 7)

String Constants

- Sequence of characters in pair of " "
- (Stored as arrays of characters in C)

```
"a string of text"
""                 /* null string */
"        "         /* string of blanks */
"   a = b + c;   " /* not an integer expression */
"  /* this is not a comment */ "
"  string with double quotes \" in it "
"  single backslash: \\ "
```

Operators and Punctuators

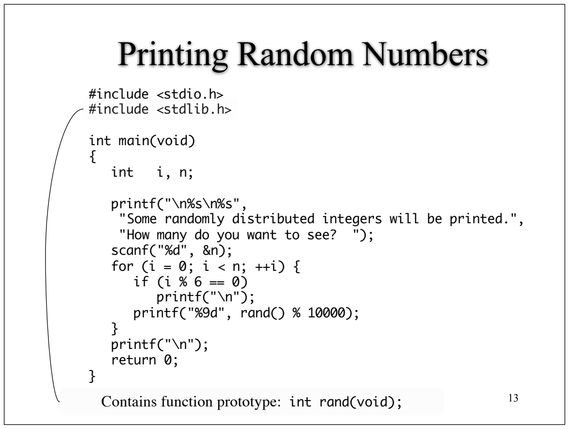
- Similar to Java
- Operators have rules of *precedence* and *associativity* (page 52)
  - `1 + 2 * 3` (* has higher precedence)
  - `1 + 2 - 3 + 4 - 5` (associate left-to-right)
  - `- a * b - c` (unary operator: higher prec.)
- Increment/Decrement operators: ++ --
  - Apply only to variables
  - Have side effects
  - Examples…

Assignment Operators

- Unlike other languages, = is a C *operator*
- Value of right side is value of the assignment expression as a whole
  - a = (b = 2) + (c = 3);
  - a = b = c = 0;
- Low precedence, associates right to left
- Other assignment operators: += -= *= /= %= >>= <<= &= ^= |=
- Examples…

Powers of Two

- Write a C program to compute the first ten powers of 2



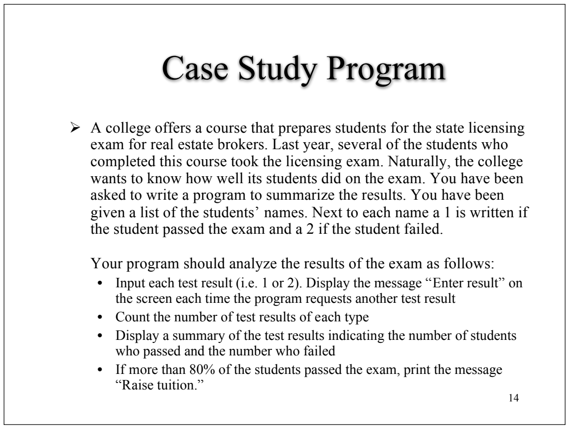
Printing Random Numbers

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   int   i, n;

   printf("\n%s\n%s",
    "Some randomly distributed integers will be printed.",
    "How many do you want to see?  ");
   scanf("%d", &n);
   for (i = 0; i < n; ++i) {
      if (i % 6 == 0)
         printf("\n");
      printf("%9d", rand() % 10000);
   }
   printf("\n");
   return 0;
}
```

Contains function prototype: `int rand(void);`



Case Study Program

- A college offers a course that prepares students for the state licensing exam for real estate brokers. Last year, several of the students who completed this course took the licensing exam. Naturally, the college wants to know how well its students did on the exam. You have been asked to write a program to summarize the results. You have been given a list of the students’ names. Next to each name a 1 is written if the student passed the exam and a 2 if the student failed.

  Your program should analyze the results of the exam as follows:
  - Input each test result (i.e. 1 or 2). Display the message “Enter result” on the screen each time the program requests another test result
  - Count the number of test results of each type
  - Display a summary of the test results indicating the number of students who passed and the number who failed
  - If more than 80% of the students passed the exam, print the message “Raise tuition.”