Basic Types and Statements

COM S 113

January 27, 1999

Read K&R sections 1.1–1.4, 1.7–1.9 by Friday.

Course web page has assignment due Monday

`http://www.cs.cornell.edu/mharris/cs113/`

Office hours: This Thursday at 11:30–12:30 in Upson B7, and Friday at 2:25–3:25 in Upson 5162

Identifiers

Consist of any number of letters, underscore, or digits, but can't start with a digit

Case sensitive

Only 31 characters of internal identifiers significant

32 reserved words—see K&R p. 192

Simple Data Types

Void type: `void`

Integral types (`signed` or `unsigned`): `char`, `short int`, `int`, `long int`

Floating-point types: `float`, `double`, `long double`

No boolean values

Integer Constants

Can be decimal, octal, or hexadecimal

Append `L` (`l`) for long integers, `U` (`u`) for unsigned

Examples:

`12`	decimal notation
`014`	decimal 12 in octal notation
`0xc`	decimal 12 in hexadecimal notation
`0XC`	same
`12L`	long constant in decimal notation

Character Constants

Escape sequences:

single quote	`\'`	new-line	`\n`
double quote	`\"`	carriage-return	`\r`
question mark	`\?`	horizontal-tab	`\t`
backslash	`\\`	vertical-tab	`\v`
alert	`\a`	arbitrary char	`\`*ooo*
backspace	`\b`	arbitrary char	`\x`*hh*
form-feed	`\f`		

Examples: `'a'` `'\0'` `'\n'` `'\\'` `'\''` `'\107'`

Floating-Point Constants

Indicated by decimal point and/or exponent

Default precision is `double`. Indicate `float` by `f` (`F`) suffix, `long double` by `l` (`L`) suffix

Examples:

```
24.0    2.4E1  2.4e1   240.0E-1
24.0F  24.0f   2.4e1f
24.0L  24.0l   2.4e1l
```

Array Types

Consists of elements of same type

Example definitions:

```
int page[10];
char line[81];
float sales[REGION][MONTHS][ITEMS];
```

Example references: `page[5]`, `line[i+j-1]`, `sales[42][11]`

String Constants

Zero or more characters enclosed in double quotes

Can include escape characters

Adjacent string constants are pasted together

`C`	`o`	`r`	`n`	`e`	`l`	`l`	`\0`
0	1	2	3	4	5	6	7

Initializers

```
float eps = 0.0001;
int i = 0, j = 0;
int year[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
float x[4] = {1, };
char thanks[25] = "Thank you for using AT&T";
char error[] = "Buffer length"    "exceeded";
float matrix[2][3] = { {1.0, 1.0, 1.0},
                       {2.0, 2.0, 2.0}, };
```

Comments

Started with `/*` and ended with `*/`

Can span lines, but can't be nested

Examples:

```
/* Cornell */ | /* /* /* Hi */ | /*Cornell...
              |                | University*/
```

Constant Identifiers

`#define` *constant-name constant-or-constant-name*

`#define` *constant-name* (*constant-expression*)

Examples:

```
#define MAX_LENGTH 100      #define EOF (-1)
#define PI 3.141592653589   #define TOTAL_ELEM (M*N)
```

Warning: Expression evaluated once per use!

Simple Statements

Null statement: denoted by semicolon

Expression statement: expression followed by semicolon. Expression is evaluated, value is discarded

Compound statement (or block)

{

definitions-and-declarations (optional)

statement-list

}

Used for grouping, as function body, and to restrict identifier visibility

Note: No semicolon after closing brace!

Example of Block Usage

```
main() {
  int i = 3;
  printf("%d ", i);
  {
    int i = 4;
    printf("%d ", i);
  }
  printf("%d\n", i);
}
```

`if` Statement

`if (`*expression*`)` *statement*$_1$

`if (`*expression*`)` *statement*$_1$ `else` *statement*$_2$

expression must have an integral type. Zero is false, nonzero is true.

Notes: (1) Expression *must* be parenthesized. (2) No `then` keyword. (3) Semicolon appears before `else` if *statement*$_1$ simple.

Dangling `else` Problem

`if (`e_1`) if (`e_2`)` s_a `else` s_b

```
if (e1)                 if (e1)
   if (e2)                 if (e2)
      sa                      sa
   else                    else
      sb                      sb
```

Avoiding the Dangling `else` Problem

Null statement

```
if (e1)
   if (e2)
      sa
   else
      ;
else
   sb
```

Compound statement

```
if (e1) {
   if (e2) sa
}
else
   sb
```

`else if` **Statements**

```
if (expression)
   statement
else if (expression)
   statement
else if (expression)
   statement
else
   statement
```

`while` Loops

`while (`*expression*`)` *statement*

```
int profit[MONTHS];
int i = 0;
while (i < MONTHS) {
  if (profit[i] > 0)
    printf("We made a profit in month %d\n", i);
  i++;
}
```

`do` Loops

`do` *statement* `while (`*expression*`);`

```
char a[] = "Cornell University", b[30];
int i = 0;
do {
  b[i] = a[i];
  i = i + 1;
} while (a[i-1] != '\0');
```

`for` Loops

`for` (*expr*$_{1opt}$`;` *expr*$_{2opt}$`;` *expr*$_{3opt}$`)` *statement*

expr$_{1opt}$ is the initialization, *expr*$_{2opt}$ is the test (is *true* if absent), *expr*$_{3opt}$ is the reinitialization.

expr$_{1opt}$`;`
`while (`*expr*$_{2opt}$`) {`
  statement
  expr$_{3opt}$`;`
`}`