C REFERENCE

A FIRST BOOK OF ANSI C by Gary Bronson and Stephen Menconi

Keywords

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

Operators

<table>
<tr><th>Type</th><th>Symbols</th><th>Associativity</th></tr>
<tr><td>Primary</td><td>( ) [ ] . -></td><td>left to right</td></tr>
<tr><td>Unary</td><td>++ -- & * - ! ~ sizeof</td><td>right to left</td></tr>
<tr><td>Arithmetic</td><td>* / %</td><td>left to right</td></tr>
<tr><td>Arithmetic</td><td>+ -</td><td>left to right</td></tr>
<tr><td>Shift</td><td><< >></td><td>left to right</td></tr>
<tr><td>Relational</td><td>< <= > >=</td><td>left to right</td></tr>
<tr><td>Relational</td><td>== !=</td><td>left to right</td></tr>
<tr><td>Bitwise</td><td>& (AND)</td><td>left to right</td></tr>
<tr><td>Bitwise</td><td>^ (XOR)</td><td>left to right</td></tr>
<tr><td>Bitwise</td><td>| (OR)</td><td>left to right</td></tr>
<tr><td>Logical</td><td>&& (AND)</td><td>left to right</td></tr>
<tr><td>Logical</td><td>|| (OR)</td><td>left to right</td></tr>
<tr><td>Conditional</td><td>?:</td><td>right to left</td></tr>
<tr><td>Assignment</td><td>= += -= /= %= etc.</td><td>right to left</td></tr>
<tr><td>Comma</td><td>,</td><td>left to right</td></tr>
</table>

Scalar Data Types

	Type	Sample Declaration
char		char key;
int		int num;
short	(or short int)	short count;
long	(or long int)	long int date;
unsigned	(or unsigned int)	unsigned val;
float		float rate;
double	(or long float)	double taxes;

Arrays

An *array* is a list of elements of the same data type. The first element in an array is referred to as the zeroth element.

Examples:

```
int prices[5];
char name[20];
float rates [4][15];
```

Structures

A *structure* (or record) is a data type whose elements need not be of the same data type.

Example:

```
struct tel_rec      /* tel_rec is a tag name */
{
  char name[20];
  int id;
  double rate;
} phone;            /* phone is a structure variable */
```

Comments

Comments are enclosed within a /* and */.

```
/* this is a sample of a comment */
```

C REFERENCE *(continued)*

Statements

A *Null* statement consists of a semicolon only.

```
;     /* the Null statement */
```

A *Simple* statement is either a Null, declaration, expression, or function statement.

```
double a;                  /* declaration statement */
taxes = rate * income;     /* an expression statement */
printf("Hello World!");    /* function statement */
```

A *Compound* statement consists of one or more statements enclosed within braces.

```
Example:  {                  /* start of compound statement */
            taxes = rate * income;
            + +count;
          }                  /* end of compound statement */
```

Flow control statements are structured statements consisting of a keyword (if, while, for, do, switch) followed by an expression within parentheses and a simple or compound statement.

```
        Statement                                   Example
if (expression)                           if (age = = 13)
  statement;                                printf("Welcome Teenager!");


if (expression)                           if (num = = 5)
  statement1;                               printf("Bingo!");
else                                      else
  statement2;                               printf("You Lose!");


if (expression)                           if (grade > = 90)
  statement1;                               printf("You got an A");
else if (expression)                      else if (grade > = 80)
  statement2;                               printf("You got a B");
      .                                   else if (grade > = 70)
      .                                     printf("You got a C");
else                                      else
  statement3;                               printf("You got a D");


switch (expression)                       switch (marcode)
{                                         {
  case value_1: statement1;                 case 1: printf("Good Morning");
  case value_2: statement2;                 case 2: printf("Good Afternoon");
        .                                   case 3: printf("Good Night");
  default: statementn;                      default: printf("Good Grief");
}                                         }


for (init; expression; alter)               for (i = 0; i < 10; + +i)
  statement;                                  printf("%d %d", i, i*i);


while (expression)                          while (num < 10)
  statement;                                {
                                              printf("number is %d",num);
                                              - -num;
                                            }


do                                          do
  statement;                                {
while (expression);                           printf("Hello");
                                              - -count;
                                            }
                                            while (count < 10);
```