Types, Operators and Expressions

Ritankar Mandal

Data Types

Type	Description	Size (bits)	Format specifier
`char`	holds one character	8	%c
`int`	holds one integer	16	%d
`float`	single-precision floating point	32	%f
`double`	double-precision floating point	64	%lf

Qualifiers: short & long

Qualifiers `short` and `long` can be applied to integers.

```
1 short int sh;
2 long int counter;
```

The sizes depend on the compiler for its own hardware, subject to the restriction that

1. `short` and `int` are at least 16 bits,
2. `long` is at least 32 bits,
3. `short` is no longer than `int`, which is no longer than `long`.

Qualifiers: signed & unsigned

1. signed variables can be positive, zero or negative. So they have the range between (-2^{n-1}) and $2^{n-1} - 1$ (in a two's complement machine), where n is the number of bits in the type. E.g., signed chars have values between -128 and 127

2. unsigned numbers are always positive or zero. So they obey the laws of arithmetic modulo 2^n, where n is the number of bits in the type. So, for instance, if chars are 8 bits, unsigned char variables have values between 0 and 255.

Declaration

All variables must be declared before use.

```
1 int lower;
2 int upper;
3 char c;
4 char line[1000];
```

You can combine all the variables of the same type to a single line.

```
1 int lower, upper;
2 char c, line[1000];
```

Declaration & Initialization

```
1 /*Declaration*/
2 char c;
3 int i;
4 /*Initialization*/
5 c = 'a';
6 i = 0;
```

A variable may also be initialized in its declaration.

```
1 char c = 'a';
2 int i = 0;
```

Qualifier: const

The qualifier `const` can be applied to the declaration of any variable to specify that its value will not be changed. For an array, the `const` qualifier says that the elements will not be altered.

The following code will generate an error (implementation-defined).

```
1 const msg[] = "Hello";
2 const int i = 10;
3 printf("i = %d\n", i);
4 i = 20; // Trying to change the value of a const variable
5 printf("i = %d\n", i);
```

Operators: Arithmetic, Relational and Logical

Binary arithmetic operators are $+, -, *, /, \%$ (the modulus operator). The % operator cannot be applied to a `float` or `double`.

Example: If a year is divisible by 4 but not by 100, or by both 100 and 400 is a leap year.

```
1 if(((year%4 == 0) && (year%100 != 0)) || (year%400 == 0))
2     printf("%d is a leap year. \n", year);
3 else
4     printf("%d is not a leap year. \n", year);
```

Relational operators are $>, >=, <, <=, ==, !=$. Logical operators are && and ||.

Relational operators have lower precedence than arithmetic operators, so $i < lim - 1$ is taken as $i < (lim - 1)$.