Types, Operators and Expressions

Data Types

Type	Size (bits)	Format specifier
char	8	%c
int	16	%d
float	32	%f
double	64	%lf

Qualifiers:

- Short
- Long

- Signed
- Unsigned

Declarations

All variables must be declared before use.

```
int lower;
int upper;
int step;
char c;
char line[1000];
```

or

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

Declarations

A variable may also be initialized in its declaration.

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

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.

const int i = 0;

const msg[] = “Hello”;

Arithmetic Operators

Binary arithmetic operators are +, -, *, /, and the modulus operator %.

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

Relational and Logical Operators

The relational operators are $> \; >= \; < \; <= \; == \; != $

Type Conversions

When an operator has operands of different types, they are converted to a common type according to a small number of rules. Automatic conversions -¿ “narrower” operand into a “wider” one without losing information.

```
/* atoi: convert s to integer */
int atoi(char s[])
{
int i, n;
n = 0;
for (i = 0; s[i] >= '0' && s[i] <= '9'; ++i)
n = 10 * n + (s[i] - '0');
return n;
}
```