C Program Structure

The Basic Block -

{

declaration of variables

executable code

}

Historically, unlike in Java and C++, *all variable declarations must precede the first line of executable code within the block.* With newer compilers this restriction may not be true, but in any case, scattering variable declarations throughout a program has *adverse effects* on the readability and maintainability of a program.

Nesting of blocks is legal and common. Each interior block may include variable declarations.

Declaration of variables

Two generic types of basic (unstructured) variable exist:

integer

floating point

Integer variables may be declared as follows:

```
char  a;       /*   8 bits                    */
short b;       /*   (usually) 16 bits      */
int   c;       /*   (usually) 32 bits      */
long  d;       /*   32 or 64 bits          */
long long  e;  /*   (usually) 64 bits      */
```

These declarations implictly create *signed* integers. An 8 bit signed integer can represent values in the range

$$[-2^7,...0,....2^7-1]$$

Signed integers are represented internally using *2's complement representaion*.

Unsigned integers

Each of these declarations may also be preceded by the qualifier *unsigned.*

```
unsigned char  a;   /*   8 bits                    */
unsigned short b;   /*   (usually) 16 bits      */
```

An 8 bit unsigned integer can represent values in the range

$$[0,....2^8-1]$$

In all modern computer systems *different hardware instructions* are used for signed and unsigned arithmetic.

Encoding of integer constants:

Integer *constants* may be expressed in several ways

decimal number	65
hexadecimal number	0x41
octal number	0101
ASCII encoded character	'A'

ALL of the above values are equivalent ways to represent the 8 bit byte whose value is:

01000001

Constants of different representation may be freely intermixed in expressions.

```
x = 11 + 'b' - 055 + '\t';

x = 0xb + 0x62 - 0x2d + 0x9;

x = 11 + 98 - 45 + 9;

x = 73;
```

Floating point data

There are two variations on floating point variables

float	32 bits
double	64 bits

Example

```
float a, b;
double c;
```

Floating point constants can be expressed in two ways

Decimal number	1024.123
Scientific notation	1.024123e+3

```
avogadro = 6.02214199e+23;
```

Executable code

Expressions

consist of (legal combinations of):

constants
variables
operators
function calls

Operators

Arithmetic:	`+, -, *, /, %`
Comparative:	`==, != , <, <=, >, >=`
Logical:	`!, &&, ||`
Bitwise:	`&, |, ~, ^`
Shift:	`<<, >>`

Special types of expression

A *statement* consists of an expression followed by a semicolon

An *assignment expression* consists of

lvalue = expression;

lvalue is short for "left-value", which in turn represents any entity that may legitimately assigned a value:

The two most common examples are:

A simple variable
A pointer dereference