Lecture 1 - PRELIMINARIES

The C character set

C uses the upper case letters A to Z, lower case letters a to z, the digits 0 to 9, and certain special characters as building blocks to form basic program elements such as constants, variables, operators, expressions.

The list of special characters

```
! * + \ * <
* ( - | { >
\ ) \ ; \ } /
` - [ : ; ?
& _ ] ' . (blank)
```

Special sequences – build as the combinations, e.g. \b \n \t and so on.

Identifiers and keywords

Identifiers are names given to various program elements, such as variables, functions, and arrays.

Identifiers consist of letters, digits and eventually the underscore sign (_) (treated as a letter) in any order, except that the first character must be a letter.

Examples of valid identifiers:

```
x y12 sum_1 _temp
names area tax_rate TABLE
```

Keywords – certain reserved words that have standard predefined meanings in C.

Standard keywords:

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

Some compilers recognize some or all of the following keywords:

```
ada far near
asm fortran pascal
entry huge
```

Data types

Typical data types

<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
<th>Typical memory requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>integer quantity</td>
<td>2 bytes or 1 word (varies from one compiler to another)</td>
</tr>
<tr>
<td>char</td>
<td>single character</td>
<td>1 byte</td>
</tr>
<tr>
<td>float</td>
<td>floating point number</td>
<td>1 word (4 bytes)</td>
</tr>
<tr>
<td>double</td>
<td>double precision floating point number (i.e. more significant figures, and an exponent which may be larger in magnitude)</td>
<td>2 words (8 bytes)</td>
</tr>
</tbody>
</table>

Constants

<table>
<thead>
<tr>
<th>Example</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>int</td>
</tr>
<tr>
<td>2345l</td>
<td>long int</td>
</tr>
<tr>
<td>2345L</td>
<td>long int</td>
</tr>
<tr>
<td>1234u</td>
<td>unsigned int</td>
</tr>
<tr>
<td>2345U</td>
<td>unsigned int</td>
</tr>
<tr>
<td>5678ul</td>
<td>unsigned long int</td>
</tr>
</tbody>
</table>

```
| 12.34    | double |
| 1e-2     | double |
| 1.2e-2   | double |
| 2.3f     | float  |
| 2.3F     | float  |
| 3.4e-2l  | long double |
| 3.4e-3L  | long double |
```

Different number systems

<table>
<thead>
<tr>
<th>Number system</th>
<th>decimal</th>
<th>octal</th>
<th>hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>31</td>
<td>037</td>
<td>0X1F</td>
</tr>
<tr>
<td>decimal</td>
<td>31</td>
<td>037</td>
<td>0X1F</td>
</tr>
</tbody>
</table>

Character constants

A character constant is a single character, enclosed in apostrophes.

```
'A' 'a' '3' '$' '
```

American Standard Code for Information Interchange (ASCII) character set

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>'A'</td>
<td>65</td>
</tr>
<tr>
<td>'x'</td>
<td>120</td>
</tr>
<tr>
<td>'3'</td>
<td>51</td>
</tr>
<tr>
<td>'$'</td>
<td>36</td>
</tr>
<tr>
<td>' '</td>
<td>32</td>
</tr>
</tbody>
</table>

Escape sequences

<table>
<thead>
<tr>
<th>Character</th>
<th>Escape sequence</th>
<th>ASCII value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bell(alert)</td>
<td>\a</td>
<td>007</td>
</tr>
<tr>
<td>backspace</td>
<td>\b</td>
<td>008</td>
</tr>
<tr>
<td>horizontal tab</td>
<td>\t</td>
<td>009</td>
</tr>
<tr>
<td>vertical tab</td>
<td>\v</td>
<td>011</td>
</tr>
<tr>
<td>newline (line feed)</td>
<td>\n</td>
<td>010</td>
</tr>
<tr>
<td>form feed</td>
<td>\f</td>
<td>012</td>
</tr>
<tr>
<td>carriage return</td>
<td>\r</td>
<td>013</td>
</tr>
<tr>
<td>quotation mark</td>
<td>&quot;</td>
<td>034</td>
</tr>
<tr>
<td>apostrophe</td>
<td>'</td>
<td>039</td>
</tr>
<tr>
<td>question mark</td>
<td>?</td>
<td>063</td>
</tr>
<tr>
<td>backlash</td>
<td>\</td>
<td>092</td>
</tr>
<tr>
<td>null</td>
<td>\0</td>
<td>000</td>
</tr>
<tr>
<td>octal escape seq.</td>
<td>\oo</td>
<td></td>
</tr>
<tr>
<td>hexadecimal . esc seq.</td>
<td>\xoo</td>
<td></td>
</tr>
</tbody>
</table>

Constant expression – expression containing only constants

```
#define MAXL 10000
#define VTAB '\013'
#define VTAB '\x7'
#define VTAB '\v'
```

String constants – string of characters enclosed in double quotation marks

```
"This is a string!"
or
** /* empty string*/
```
Technically string is a table with a number of elements greater of one than the number of characters included. The characters are followed by the null character (\0).

Constant 'x' is not equal to "x".

Character strings may be stuck together during compilation of the program:

"Hey!" "Adventure!"
is identical with:
"Hey!Adventure!"

Enumeration constants
(List of values of integer constants)

enum boolean {NO,YES};
enum escapes {BELL='\a', BACKSPACE='\b', TAB='\t',
NEWLINE='\n', VTAB='\v', RETURN='\r'};
enum months {JAN=1, FEB, MAR, APR, MAY, JUN, JUL,
AUG, SEP, OCT, NOV, DEC};
/* months: february is second, march third
and so on */

Names in different enumerations must be different. At the same enumeration values may be repeated.

Types name enum shares the same space as the names of structure and union types.

Names of enumeration variables belongs to the same class as the identifiers of the ordinary variables.

Declarations

int lower, upper, step;
char c, lin[1000];

int lower;
int upper;
int step;
char c;
char lin[1000];

Initial values can be assigned to variables within a type declaration:

char esc='\\'; /* \ character */
int i=0; /* iterations counter */
int limit=MAXLINE+1; /* maximal number of iterations*/
float eps=1.0e-5; /* accuracy parameter */

Default initial values of variables:

static and outer -- 0

automatic -- when the initial values are explicitly stated the same value with call of the function or entry to the block.
without explicitly stated initial value they have random values.

Qualifier const (constant) (may be used to declare any variable) It says that its value shall not be changed.

Any trial to change value of a variable declared as constant is ended in the way depending on an implementation.

char text[]="California";

An 11 - element array.


Size should be specified correctly.

char text[6]="California"; /* end will be lost */
char text[20]="California"; /* extra array elements may be assigned zeros, or may be filled with meaningless characters */

Expressions

An expression represents a single data item, such as a number or a character. The expression may consist of a single entity, such as a constant, a variable, an array element or a reference to a function. It may also consist of some combinations of such elements interconnected by one or more operators.

Expressions can also represent logical conditions (in C true is represented by integer 1, false by 0).

Statements

A statement causes computer to carry out some action. There are three different classes of statements in C: expression statements, compound statements and control statements.

Expression statement -- expression followed by a semicolon.

a=3;
c=a+b;
++i;
printf("Area = %f", area);

Compound statement -- several individual statements enclosed within a pair of braces ( { and })

{ 
  pi=3.141593;
circum=2. * pi * radius;
area = pi * radius * radius;
}

Control statements

while (count <= n) {
  printf("x= ");
  scanf("%f", &x);
  sum += x;
  ++count;
}