C

- C konstruerades i början på sjuttiotalet av Dennis Ritchie vid Bell Laboratories.
- För att användas vid implementering av UNIX.
- Bygger på de tidigare språken BCPL och B.
- Målsättning: Att få ett högnivåspråks fördelar men ändå ha kvar kontakten med Maskinvaran

- Första ”standard”: Appendix i boken *The C Programming Language*, Dennis Ritchie och Brian Kernighan, Prentice-Hall, 1978. Kallas *K&R C*

- ANSI-standard, 1989. Kallas *ANSI-C*

- Reviderad ANSI-standard 1999. Kallas *C99*.

Språk påverkade av C:

- C++
- Java
- C#
- Objective-C
- Ett antal skriptspråk, t.ex. Perl

Det första programmet

Filen ex1.c

```c
#include <stdio.h>
int main()
{
  printf("Hello!\n");
  return 0;
}
```

```
> gcc ex1.c
> gcc -o hello.exe ex1.c
> hello.exe
Hello!
>
```

Programutvecklingsystem:

- Bara gcc

  Allmänt: gcc.gnu.org. Speciellt för Windows: www.mingw.org

  Fungerar fint tillsammans med Notepad++, notepad-plus-plus.org

- CodeLite (gcc ingår)

  www.codelite.org

- Dev-C++ (enkelt intuitivt, gcc ingår)

  www.bloodshed.net/dev/devcpp.html

- Eclipse

  www.eclipse.org/downloads

- Visual Studio

  web.student.chalmers.se/msdnaa/elms_login.php?action=signin

- Code::Blocks, Gnat, NetBeans, …

Reserverade ord:

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

Variabler:

```
char teck;
int tal;
teck = ’x’;          // tilldelning
tal  = 5;
char teck = ’A’;     // initiering
int  tal  = 0;

float maxtal = max(a, b);

int i, j, k;

int i = 1, j = 2, k = 3;

int m, n = 5;

const char slut = ’\0’;
const int  max_antal = 1000;
const int  storlek = n * k;
const antal_element = 50;    // varning i C99
const grans_1 = 28, grans_2 = 48, grans_3 = 60; //varning i C99
```

Heltalstyper:

```
_Bool                       minst  1 bit
signed char                 minst  8 bitar
unsigned char               minst  8 bitar
[signed] short [int]        minst 16 bitar
unsigned short [int]        minst 16 bitar
[signed] int                minst 16 bitar
unsigned [int]              minst 16 bitar
[signed] long [int]         minst 32 bitar
unsigned long [int]         minst 32 bitar
long long int               minst 64 bitar
unsigned long long int      minst 64 bitar
```

I C99 även: `uint_32_t, int_least16_t, int_fast32_t,` etc.

Konstanter (konstanta värden i programkoden), s.k. *literaler*:

```
4    33    9    100000                  // decimala konstanter
01   077    02653271                    // oktala konstanter
0x1    0X75    0Xff    0xD7A02          // hexadecimala konstanter
```

Heltalskonstanter får normalt typen **`int`**. Kan ändras:

```
45L          får typen long int
0653U        får typen unsigned int
0x4a65l      får typen long int
0501u        får typen unsigned int
0775ul       får typen unsigned long int
8239775LL    får typen long long int
```

Kodning av skrivbara symboler (tecken):

- ASCII, 7 bitar, 128 tecken, bokstäver a-z, siffror, vanliga specialtecken
- Latin_1, 8 bitar, 256 tecken, ASCII + västeuropeiska bokstäver + fler specialtecken
- Unicode, max 24 bitar, max 1 114 112 tecken från alla väldens hörn

  Varje tecken har ett unikt nummer, en s.k. *code point*

  Tecken nummer 0-127 har samma kod som i ASCII

  Tecken som kan kodas med max 16 bitar ingår i *Basic Multilingual Plane* (*BMP*)

  UTF_32, 32 bitar används för varje tecken

  UTF_16, ett eller två 16-bitars ord används för varje tecken

  UTF_8, ett, två eller tre 8-bitars ord (bytes) används för varje tecken, kallas *multibyte character* i C

Teckenkonstanter (får typen `int`):

```
'a'   'F'   '>'   '7'
```

Escape-sekvenser:

`\'`	enkelapostrof
`\"`	dubbelapostrof
`\?`	frågetecken
`\\`	tecknet \

`\0` (*null-character*) Ett tecken där alla bitar har värdet 0.

`\a` (*alert*) Ger en ljud- eller ljussignal på terminalen.

`\b` (*backspace*) Flyttar utskriftspositionen ett steg till vänster.

`\f` (*form feed*) Flyttar utskriftspositionen till början på nästa sida.

`\n` (*new line*) Flyttar utskriftspositionen till början på nästa rad.

`\r` (*carriage return*) Flyttar utskriftspositionen till början på aktuell rad.

`\t` (*horisontal tab*) Flyttar fram utskriftspositionen till nästa tabstopp.

`\v` (*vertical tab*) Flyttar utskriftspositionen till början på den rad som är markerad som nästa vertikala tabulatorstopp.

```
'\5'   '\53'   '\160'   '\377'     // oktal teckenkod
'\x9'   '\xd'   '\x5b'   '\x3ff'   // hexadecimal teckenkod
```

I C99: `wchar_t` (16 eller 32 bitar) kan representera alla tecken som ingår i BMP i Unicode

```
wchar_t cw =  L'\u03B1';          /* lilla alfa */
```

I standarförslag: `char16_t` och `char32_t`

```
char16_t c16 = u'\u20AC';          /* euro-tecknet */
char32_t c32 = U'\U0001D121';      /* C-klav */
```

Textsträngsliteraler där varje tecken ligger i en **`char`**:

```
"A string literal"

printf("Rad 1\nRad 2\tJag piper nu\a\n");
"C:\\Cpp3\\Ex\\kap3\\temp.txt"
"Han ropade: \"Stopp!\""
```

Textsträngsliteraler där varje tecken ligger i en `wchar_t`:

```
  L"Price: 10\u20AC"       /* 20AC är koden för euro-tecknet */
```

Flyttalstyper:

float
double
long double

Flyttalskonstanter (får typen **double**):

```
567.7       0.0041       5.0       57.
  0.3        .003         .0        0.
```

Kan ha exponentdel:

```
0.98765e15       3.11E-5       1.e+12
 .543E-17        249e25        0e0
```

Flyttalskonstanter får normalt typen **double**. Kan ändras:

```
7.45f       0.624e-7f       0.0F          // får typen float
7.123456768656574L      0.93432e50L       // får typen long double
```

```
long double             högsta typen
double
float
unsigned long long int
long long int
unsigned long int
long int
unsigned int
int
unsigned short int
short int
unsigned char
signed char
_Bool                   lägsta typen
```

Automatiska typomvandlingar

- Operanderna till en operator kan omvandlas.
- Vid tilldelning omvandlas uttrycket i högerledet till den typ som operanden till vänster har.
- En aktuell parameter till en funktion kan ibland omvandlas. (Se kap 6.)
- Värdet som returneras från en funktion omvandlas till den typ som funktionen skall ge som resultat.

Vid operatorer: *de vanliga aritmetiska typomvandlingarna*.

Den operand som har ”lägst” typ omvandlas till den andra operandens typ.

Bivillkor: lägst typen **int**.

Explicita typomvandlingar (casts)

```
(typnamn) uttryck
```

```
(unsigned int) i
(long double) (x + y)
(char) k
```

Tillåtet för aritmetiska typer och pekartyper

`printf`, utskrift av heltal:

`d`	Värdet tolkas som **`int`**, utskrift på decimal form
`i`	Samma som d
`u`	Värdet tolkas som **`unsigned`** `int`, decimal utskrift
`o`	Värdet tolkas som **`unsigned`** `int`, oktal utskrift
`x`	Värdet tolkas som **`unsigned`** `int`, hexadecimal utskrift
`X`	Samma som x
`ld`	Värdet tolkas som **`long`** `int`, decimal utskrift
`li`	Samma som ld
`lu`	Värdet tolkas som **`unsigned`** `long int`, decimal utskrift
`lo`	Värdet tolkas som **`unsigned`** `long int`, oktal utskrift
`lx`	Värdet tolkas som **`unsigned`** `long int`, hexadecimal utskrift
`lX`	Samma som `lx`
`ll`	Kan stå framför istället för bara `l`. Man får då **`long`** `long int`, `unsigned long long int`, etc

`printf`, utskrift av flyttal:

`f,lf` Värdet tolkas som **`double`**. Utskrift sker på ”vanlig” form med minst en heltalssiffra.
Antalet decimaler bestäms av precisionsangivelsen. Om precision inte anges skrivs 6 decimaler ut.

`e,le` Värdet tolkas som **`double`**. Utskrift sker på exponentform med en heltalssiffra.
Antalet decimaler bestäms av precisionsangivelsen. Om precision inte anges skrivs 6 decimaler ut.

`E,lE` Samma som e men med den skillnaden att ett stort E används i stället för ett litet e i utskriften.

`g,lg` Värdet tolkas som **`double`**. Utskrift sker antingen på ”vanlig” form eller på exponentform.
Om värdet är ”stort” (exponenten är större än precisionen) eller ”litet” (exponenten är mindre än -4) sker utskriften på exponentform, annars på ”vanlig” form.

`G,lG` Samma som formen g men med den skillnaden att ett stort E
används i stället för ett litet e om utskriften sker på exponentform.

`Lf` Samma som formen `f` men värdet tolkas som **`long double`**.

`Le` Samma som formen `e` men värdet tolkas som **`long double`**.

`LE` Samma som formen `E` men värdet tolkas som **`long double`**.

`Lg` Samma som formen `g` men värdet tolkas som **`long double`**.

`LG` Samma som formen `G` men värdet tolkas som **`long double`**.

`printf`, utskrift av text:

`c`	en **`char`** innehållande en teckenkod, motsvarande tecken skrivs ut
`s`	en pekare till ett fält med **`char`** innehållande teckenkoder, motsvarande text skrivs ut
`lc`	en `wchar_t` innehållande en teckenkod, motsvarande tecken skrivs ut
`ls`	en pekare till ett fält med `wchar_t` innehållande teckenkoder, motsvarande text skrivs ut

`scanf`, inläsning av heltal:

`hd`	De inlästa tecknen tolkas som ett decimalt heltal och placeras i en variabel av typen **`short int`**.
`hu`	De inlästa tecknen tolkas som ett decimalt heltal och placeras i en variabel av typen **`unsigned short int`**.
`d`	De inlästa tecknen tolkas som ett decimalt heltal och placeras i en variabel av typen **`int`**.
`u`	De inlästa tecknen tolkas som ett decimalt heltal och placeras i en variabel av typen **`unsigned int`**.
`ld`	De inlästa tecknen tolkas som ett decimalt heltal och placeras i en variabel av typen **`long int`**.
`lu`	De inlästa tecknen tolkas som ett decimalt heltal och placeras i en variabel av typen **`unsigned long int`**.
`ho`	De inlästa tecknen tolkas som ett oktalt heltal och placeras i en variabel av typen **`short int`**.
`o`	De inlästa tecknen tolkas som ett oktalt heltal och placeras i en variabel av typen **`int`**.
`lo`	De inlästa tecknen tolkas som ett oktalt heltal och placeras i en variabel av typen **`long int`**.
`hx`	De inlästa tecknen tolkas som ett hexadecimalt heltal och placeras i en variabel av typen **`short int`**
`x`	De inlästa tecknen tolkas som ett hexadecimalt heltal och placeras i en variabel av typen **`int`**.
`lx`	De inlästa tecknen tolkas som ett hexadecimalt heltal och placeras i en variabel av typen **`long int`**.
`ll`	istället för `l` betyder att typen blir **`long long int`**, etc

`scanf`, inläsning av flyttal:

`e, f, g`	De inlästa tecknen skall tolkas som ett reellt tal och placeras i en variabel av typen **`float`**.
`le, lf, lg`	De inlästa tecknen skall tolkas som ett reellt tal och placeras i en variabel av typen **`double`**.
`Le, Lf, Lg`	De inlästa tecknen skall tolkas som ett reellt tal och placeras i en variabel av typen **`long double`**.

`scanf`, inläsning av tecken och texter:

`c` en pekare till en **`char`**, ett tecken läses in

`s` en pekare till ett fält med **`char`** innehållande, ett *ord* läses in, nolltecken läggs till

`[`*`text`*`]` som `s`, men matchar bara tecken som ingår i *`text`*

`^[`*`text`*`]` som `s`, men matchar bara tecken som inte ingår i *`text`*

`l` framför, som ovan men för `wchar_t`, inlästa tecken tolkas som multibyte characters

```
typedef float vikt, volym;
typedef unsigned char text[100];

volym vol1, vol2;
vikt vi1, vi2, vi3;
text if1, if2;


typedef unsigned short int wchar_t;   /* tänkbar definition av wchar_t */
```