Historia języka C

1969 język BCPL; Martin Richards z University Mathematical Laboratories w Cambridge; dla pierwszej instalacji systemu operacyjnego UNIX

1970 język B zdefiniowany dwa lata wcześniej przez Kena Thompsona dla pierwszego systemu UNIX działającego na DEC PDP-7.

1972 definicja języka C przez Dennisa M. Ritchie z Bell Laboratories w New Jersey dla systemu UNIX działającego na minikomputerze DEC PDP-11.

Cele:

- połączenie cech języka wysokiego poziomu zapewniając łatwość projektowania algorytmów, a zarazem dostatecznie niskiego, aby algorytmy te były optymalne.
- uniezależnienie takiego języka od konkretnej maszyny i systemu operacyjnego, przy uwzględnieniu jednocześnie właściwości jak największej liczby maszyn.

Zestaw znaków języka C

C używa liter dużych A do Z, liter małych a do z, cyfr 0 do 9 i pewnych znaków specjalnych do budowy swoich podstawowych elementów programu takich, jak stałe, zmienne, operatory i wyrażenia.

Lista znaków specjalnych

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

Ciągi specjalne – budowane jako kombinacje, np. `\b \n \t` iotd.

Identyfikatory i słowa kluczowe

Identyfikatorami są nazwy nadawane różnym elementom programu takim, jak zmienne, funkcje i tablice.

Identyfikatory składają się z liter, cyfr oraz ewentualnie znaku podkreślenia (_) (traktowanego jak litera) występujących w dowolnym porządku, z jednym ograniczeniem, że pierwszym znakiem musi być litera.

Przykłady prawidłowych identyfikatorów:

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

Słowa kluczowe – pewne zarezerwowane wyrazy, które mają predefiniowane znaczenie w C.

Standardowe słowa kluczowe:

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

Pewne kompilatory rozpoznają wszystkie albo część następujących słów kluczowych:

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

Typy danych

Typowe typy danych

Typ danych	Opis	Typowe wymagania pamięciowe
int	wielkość całkowita	2 bajty albo 1 słowo (zmienia się zależnie od kompilatora)
char	pojedyńczy znak	1 bajt
float	liczba zmienno-przecinkowa	1 słowo (4 bajty)
double	liczba zmiennoprzecinkowa podwójnej precyzji (tzn. najbardziej znaczące cyfry i wykładnik, które mogą mieć większy rozmiar)	2 words (8 bytes),

Stałe

Przykład	Nazwa
1234	int
2345l	long int
2345L	long int
1234u	unsigned int
2345U	unsigned int
5678ul	unsigned long int
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

Różne systemy liczbowe

System liczbowy		
dziesiętny	ósemkowy	szesnastkowy
31	037	0x1f
31	037	0X1F

Stałe znakowe

Stała znakowa jest pojedyńczym znakiem zamkniętym pomiędzy apostrofami.

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

Zestaw znaków ASCII (American Standard Code for Information Interchange)

Stała	Wartość
'A'	65
'x'	120
'3'	51
'$'	36
' '	32

Znaki specjalne

Znak	Symbol	Wartość ASCII
bell(alarm)	\a	007
cofnięcie	\b	008
poziomy tabulator	\t	009
pionowy tabulator	\v	011
nowa linia (line feed)	\n	010
nowa strona (form feed)	\f	012
powrót karetki	\r	013
cudzysłów	\"	034

```
apostrof              \'              039
znak zapytania        \?              063
backslash             \\              092
null                  \0              000
liczba ósemkowa        \ooo
liczba szesnastkowa   \xoo
```

Wyrażenie stałe – wyrażenie zawierające tylko stałe

```
#define MAXL 10000
#define VTAB '\013'
#define VTAB '\x7'
#define VTAB '\X7'
#define VTAB '\v'
```

Stałe napisowe – ciągi znaków zamknięte pomiędzy znakami cudzysłowu

```
"To jest napis!"
albo
"" /* napis pusty*/
```

Technicznie napis jest tablicą o liczbie elementów większej o jeden od liczby znaków, które zawiera napis. Po znakach występuje znak NULL (`\0`).

Stała `'x'` jest różna od `"x"`.

Stałe napisowe mogą być połączone razem w trakcie kompilacji programu:

```
"Hej!" "Przygodo!"
jest identyczne z:
"Hej!Przygodo!"
```

Stałe wyliczeniowe

(Lista wartości stałych całkowitoliczbowych).

```
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  */
```

Nazwy w różnych wyliczeniach muszą być różne. Wartości w tym samym wyliczeniu mogą się powtarzać.

Nazwa typu *enum* należy do tej samej przestrzeni co nazwy typów struktur i unii.

Nazwy zmiennych wylioczeniowych należą do tej samej klasy, co identyfikatory zwykłych zmiennych.

Deklaracje

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

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

Wartości początkowe mogą być przypisane zmiennym wewnątrz deklaracji typu:

```
char esc='\\';   /*  znak \ */
int  i=0;        /*  licznik iteracji    */
int  limit=MAXLINE+1; /* maks. liczby iteracji  */
float eps=1.0e-5;     /* parametr dokładności       */
```

Domyślne wartości początkowe zmiennych:

```
statyczne i zewnętrzne -- 0
automatyczne           -- gdy wartości początkowe są
                          określone ta sama wartość
                          przy każdym wywołaniu funk-
                          cji albo wejściu do bloku.
                          Bez określenia wartości po-
                          czątkowych przyjmują one
                          wartości losowe.
```

Kwalifikator *const* (constant) (może być użyty do zadeklarowania dowolnej zmiennej) Mówi, że wartość zmiennej nie będzie zmieniona.

```
const double e=2.1234e-2;
const char msg[]="Uwaga";
int strlen(const char []);
/* zawartość tablicy -- argument nie może
     być zmieniony wewnątrz funkcji */
```

Każda próba zmiany wartości zmiennej zadeklarowanej jako constant kończy się w sposób zależny od implementacji.

```
char text[]="Kalifornia";
```

Tablica 11 - elementowa.

```
char text[11]="Kalifornia";
```

Rozmiar powinien być wyspecyfikowany właściwie.

```
char text[6]="Kalifornia";  /*  Stracimy koniec   */
char text[20]="Kalifornia"; /*  dodatkowe elementy
             mogą być wyzerowane, albo mogą zostać
             wypełnione znakami bez znaczenia */
```

Wyrażenia

Wyrażenie reprezentuje pojedyńczą jednostkę danych, tak samo jak liczba albo znak. Wyrażenie może składać się z pojedyńczej wielkości takiej jak stała, zmienna, element tablicy albo odniesienie do funkcji. Może również składać się z pewnych kombinacji takich elementów połączonych wzajemnie przez jeden albo więcej operatorów.

Wyrażenie może równiez reprezenrtiować warunek logiczny (w C prawda jest reprezentowana przez liczbę całkowitą 1 a fałsz przez 0).

Instrukcje

Instrukcja poleca komputerowi wykonać pewną czynność. W C występują trzy różne klasy instrukcji: *instrukcje wyrażenia, instrukcje blokowe i instrukcje sterujące.*

Instrukcje wyrażenia – wyrażenia kończone znakiem średnika (;).

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

Instrukcja blokowa – grupa instrukcji pojedyńczych otoczonych parą nawiasów klamrowych ({ and })

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

Instrukcje sterujące

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