

Język ANSI C

Pierwsze starcie.

ZNOWU TROCHĘ HISTORII



- 1972 Dennis Ritchie (Bell Labs., New Jersey), projekt języka C na bazie języka B
- 1973 UNIX, jądro w C, pierwszy przenośny system operacyjny
- 1978 D. Ritchie, Brian Kernighan, „*The C Programming Language*”
- 1983 Bjarne Stroustrup, Język C++
- 1989 Standard ANSI C, standardowe C, „pure C”, C89, C90
- 1999 Standard C99
- 2011 Standard C11

STRUKTURA PROGRAMU W C

```
#include <stdio.h>          /* Dyrektywy preprocesora */
#define PI 3.1415

int main()                  /* Funkcja glowna */
{
    int i;                  /* Deklaracje */
    float x;

    i = 10 * PI;            /* Instrukcje */
    x = 1.0;

    return 0;
}
```

NAJKRÓTSZY PROGRAM

Najkrótszy program w C

```
main() {}
```

Witaj świecie

```
#include<stdio.h>

int main()
{
    puts("Witaj świecie!");
    return 0;
}
```

- mały język ale duże możliwości, ważna rola bibliotek
- pliki źródłowe *.c
- pliki nagłówkowe *.h, zawierają deklaracje typów i funkcji, nie zawierają instrukcji
- biblioteki: zbiory typów danych, stałych i funkcji
- modularność: programowanie proceduralne
- dostęp do pamięci i rejestrów
- zwięzły kod - ale nie wolno przesadzać
- C nie chroni programisty przed nim samym

OGÓLNE O PROCESIE KOMPILACJI

1 preprocesor

pliki źródłowe (*.c, *.h) $\Rightarrow$ przetworzone pliki
dyrektywy preprocesora, np.:

```
#include<stdio.h>
```

```
#define PI 3.14
```

2 kompilacja

przetworzone pliki $\Rightarrow$ pliki obiektowe (*.obj, *.o)

3 konsolidacja (linkowanie)

pliki obiektowe + biblioteki $\Rightarrow$ program (*.exe , a.out)

BINARNA REPREZENTACJA ZMIENNYCH

- zmienna zajmuje pamięć: 1B, 2B, 4B, 8B, ...
- adres zmiennej w pamięci:
- operacje na bitach
- typy całkowite ze znakiem i bez znaku, skończony zakres
- typy zmiennoprzecinkowe: rozdzielczość i skończoność

`char a='C';`

`&a`

0	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

 = 67

PODSTAWOWE TYPY DANYCH

- float - zmiennoprzecinkowa (*floating point*),
4 bajty
zakres $[-3.4 \times 10^{38}, 3.4 \times 10^{38}]$,
wartość minimalna większa od zera 1.17×10^{-38}
- int - liczby całkowite (*integer*)
4 bajty
zakres $[-2^{31}, 2^{31}]$
- char - znak (*character*)
1 bajt
zakres $[-127, 128]$, kod ASCII
- brak typu logicznego
0 to fałsz, nie 0 to prawda
- void - typ pusty, brak typu
- unsigned int - całkowita bez znaku

DEKLARACJA

powiązanie zmiennej, stałej lub funkcji danego typu z identyfikatorem (unikatową nazwą).

IDENTYFIKATORY

- litery a-z, A-Z, cyfry 0-9 (ale nie pierwsza), podkreślnik _
- małe i duże litery rozróżniane: zmienna, Zmienna, ZMIENNA

Przykłady deklaracji zmiennych:

```
int a,b,c;  
float srednica_kola;  
char PewienZnak;  
float x1, x2, x3;
```

Niepoprawne nazwy: 2abc, promień, wazna zmienna,
pewna-zmienna

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

INSTRUKCJA PROSTA

wyrażenie ;

```
a = x + 1;
```

Instrukcja prosta jest zawsze zakończona średnikiem.

INSTRUKCJA ZŁOŻONA (BLOK INSTRUKCJI)

```
{  
    instrukcja 1  
    instrukcja 2  
    instrukcja 3  
}
```

```
{  
    float x, y;  
    x = 1.0;    y = 2.4;  
    x = x + y;  
}
```

INSTRUKCJE STERUJĄCE

if else do while

OPERATORY ARYTMETYCZNE

* / % + -

OPERATORY RELACJI

< > <= >= == !=

OPERATORY LOGICZNE

! && ||

OPERATOR PRZYPISANIA

=

OPERACJA PRZYPISANIA WARTOŚCI

zmienna = wyrażenie;

```
int i, j, k;           /* deklaracja */
float x = 1.5;         /* inicjalizacja */
float y = 1.5e-5;
char znak = 'A';

i = i + 3;             /* brak inicjalizacji */
3 = x;                 /* złe */
x + y = x - y;         /* złe */
znak = 65;
```

	przypisanie	porównanie
C	$a = 3$	$a == 3$
Pascal	$a := 3$	$a = 3$
pseudo-kod	$a \leftarrow 3$	$a = 3$

OPERATORY ARYTMETYCZNE

OPERATORY ARYTMETYCZNE

*	/	%	wyższy priorytet
+	-		niższy priorytet

Reszta z dzielenia (modulo, %) tylko dla zmiennych całkowitych

Dzielnie bez reszty (/) dla zmiennych całkowitych

KOLEJNOŚĆ OBLICZEŃ

$$z = \frac{x}{2y}$$

```
z = x / 2 * y;      /* złe */  
z = x / (2 * y);
```

$$z = \frac{x - y}{x + y}$$

```
z = x - y / x + y;  /* złe */  
z = (x - y) / (x + y);
```

Obliczenia od lewej do prawej, priorytet * / % przed + -

PRZYKŁAD: POLE I OBWÓD KOŁA

```
1  #include <stdio.h>
2  #define PI 3.14159
3
4  int main()
5  {
6      float r,pole,obw;
7
8      printf("Podaj promien kola\nr= ");
9      scanf("%f",&r);
10
11     pole = PI*r*r;
12     obw  = 2*PI*r;
13
14     printf("Pole kola o promieniu %f wynosi %f\n",r,pole);
15     printf("Obwod kola o promieniu %f wynosi %f\n",r,obw);
16
17     return 0;
18 }
```

```
#include<stdio.h>
```

stdio.h - **S**tandard input/output, biblioteka obsługująca komunikację ze standardowym wejściem i wyjściem aplikacji.

FUNKCJA PRINTF()

formatowane wyjście (wyświetlanie w terminalu)

```
printf("format", arg1, arg2, ... )
```

FUNKCJA SCANF()

formatowane wejście (odczyt z klawiatury)

```
scanf("format", adres1, adres2, ... )
```

SPECYFIKACJA FORMATU

FORMAT

%f	zmiennoprzecinkowa	printf("%f",3.14);	3.140000
%.2f	2 miejsca po przecinku	printf("%.2f",3.14);	3.14
%e	notacja naukowa	printf("%e",0.01);	1.000000e-02
%d	dziesiętne	printf("%d",75);	75
%c	znak	printf("%c",75);	K
%x	szesnastkowo	printf("%x",75);	4b
%s	łańcuch znakowy	printf("%s","napis");	napis

SYMBOLE SPECJALNE

\n	nowa linia	\\	\ <i>backslash</i>
\t	tabulator	\' \" \?	' " ?
\b	backspace	\r	powrót karetki
%%	%		

SCANF - KILKA WAŻNYCH UWAG

- dopasowanie formatu do typu zmiennej

```
int a;  
scanf("%f",&a);                                /* problem !!!*/
```

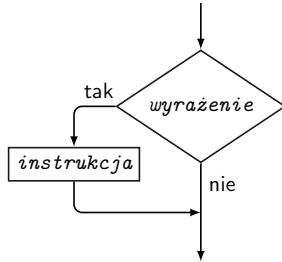
- argumentem jest adres zmiennej (pamiętaj o &)
- formaty liczbowe %f i %d zjadają poprzedzające białe znaki
- spacja w formacie pomija wszystkie białe znaki
- znak niepasujący do formatu przerywa wczytywanie i pozostaje w strumieniu wejściowym

```
char a;  
float x,y;  
scanf("%f%f",&x,&y);                            /* wczytanie 2 liczb */  
scanf("%c",&a);                                /* czyta znak */  
scanf(" %c",&a);                               /* pomija białe znaki */
```

INSTRUKCJA WARUNKOWA IF ELSE

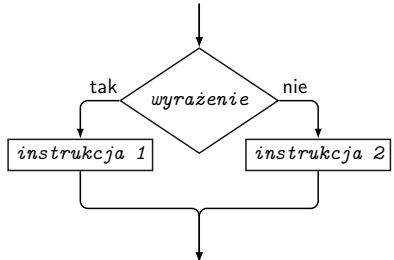
WARUNEK IF (JEŻELI)

```
if ( wyrażenie )  
    instrukcja
```



WARUNEK IF ELSE

```
if ( wyrażenie )  
    instrukcja 1  
else  
    instrukcja 2
```



PRZYKŁAD: RÓWNANIE Z JEDNĄ NIEWIADOMĄ

Problem: znajdź miejsce zerowe funkcji liniowej

$$f(x) = ax + b$$

Algorytm 1 Równanie z jedną niewiadomą

Dane wejściowe: współczynniki $a, b \in \mathbb{R}$

Wynik: miejsce zerowe $x_0 \in \mathbb{R}$ lub informacja o braku rozwiązania

- 1: **jeżeli** $a \neq 0$ **wykonaj**
 - 2: $x_0 \leftarrow -\frac{b}{a}$
 - 3: **wypisz:** x_0
 - 4: **w przeciwnym wypadku**
 - 5: **wypisz:** Brak rozwiązania
-

```

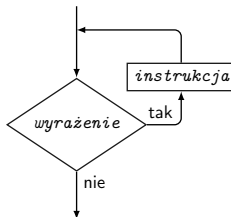
1  #include <stdio.h>
2
3  int main()
4  {
5      float a, b, x0;
6
7      printf("Podaj współczynniki równania\n");
8      printf("a = "); scanf("%f",&a);
9      printf("b = "); scanf("%f",&b);
10
11     if( a != 0.0 )
12     {
13         x0=-b/a;
14         printf("x0 = %.4f\n",x0);
15     }
16     else printf("Brak rozwiązań\n");
17
18     return 0;
19 }

```

PĘTLA WHILE I DO WHILE

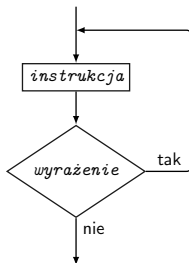
PĘTLA WHILE (DOPÓKI)

```
while ( wyrażenie )  
    instrukcja
```



PĘTLA DO WHILE

```
do  
    instrukcja  
while ( wyrażenie );
```



PRZYKŁAD: WYZNACZANIE SILNI $n!$

Problem: wyznaczenie wartości silni $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$

Algorytm 2 Silnia

Dane wejściowe: liczba całkowita $n \geq 0$

Wynik: wartość $x = n!$

- 1: $i \leftarrow 2$
 - 2: $x \leftarrow 1$
 - 3: **dopóki** $i \leq n$ **wykonuj**
 - 4: $x \leftarrow x \cdot i$
 - 5: $i \leftarrow i + 1$
 - 6: **wypisz** x
-

```
1  #include<stdio.h>
2
3  int main()
4  {
5      int x, i, n;
6
7      printf("n = ");  scanf("%d",&n);
8
9      if(n<0)
10     {
11         printf("Zle dane: n<0\n");
12     }
13     else
14     {
15         i=2;  x=1;
16         while( i <= n )
17         {
18             x = x * i;
19             i = i + 1;
20         }
21         printf("%d! = %d\n",n,x);
22     }
23     return 0;
24 }
```

PRZYKŁAD: ALGORYTM HERONA

METODA BABILOŃSKA

Problem: wyznaczenie wartości pierwiastka $\sqrt{a}$

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right) \xrightarrow{n \rightarrow \infty} \sqrt{a}$$

Algorytm 3 Algorytm Herona

Dane wejściowe: liczba $a \geq 0$, wartość początkowa $x_0 > 0$, dokładność obliczeń $\epsilon > 0$

Wynik: wartość przybliżona $x \approx \sqrt{a}$ z dokładnością ϵ

- 1: $x \leftarrow x_0$
 - 2: **wykonuj**
 - 3: $x_0 \leftarrow x$
 - 4: $x \leftarrow \frac{1}{2} \left(x_0 + \frac{a}{x_0} \right)$
 - 5: **dopóki** $|x - x_0| > \epsilon$
 - 6: **wypisz** x
-

```

1  #include<stdio.h>
2
3  int main()
4  {
5      const float eps=1e-4;
6      float x,a,x0;
7
8      printf("a = "); scanf("%f",&a);
9
10     if( a < 0 ) printf("Zle dane: a < 0\n");
11     else
12     {
13         x0 = 1;
14         x = x0;
15         do
16         {
17             x0 = x;
18             x = (x0 + a/x0)/2;
19         }while(x - x0 > eps || x - x0 < -eps);
20
21         printf("pierwiastek z %f wynosi %f (eps= %f)\n",a,x,eps);
22     }
23     return 0;
24 }

```

BĄDŹ KOMPILATOREM

WYPISYWANIE PODZIELNIKÓW LICZBY CAŁKOWITEJ

```
1  #include<studio.h>;
2
3  char main()
4  {
5      int n
6
7      printf("Podaj liczbe calkowita wieksza od zera: ");
8      scanf("%f",&n);
9
10     if(n <= 0)
11         if ( n==0 ) printf("To jest zero!\n");
12     else
13     {
14         printf("Dzielniki liczby %d:\n ",n);
15         int i;
16         while( i<n );
17         {
18             if( n % i ) printf("%c/n",i);
19             i = i + 1;
20         }
21     }
22     return 0;
23 }
```

BĄDŹ KOMPILATOREM

```
1  #include<stdio.h>                                /* studio.h, srednik */
2
3  int main()                                         /* int */
4  {
5      int n,i=1;                                    /* srednik , deklaracja , inicjalizacja */
6
7      printf("Podaj liczbe calkowita wieksza od zera : ");
8      scanf("%d",&n);                                /* format , adres */
9
10     if(n <= 0)
11     {
12         if ( n==0 ) printf("To jest zero!\n");      /* nawiasy */
13     }
14     else
15     {
16         printf("Dzielniki liczby %d:\n",n);
17         while( i<=n )                                /* srednik , p. nieskonczona */
18         {
19             if( n % i == 0 ) printf("%d\n",i);        /* %d, \n, == */
20             i = i + 1;
21         }
22     }
23     return 0;
24 }
```

STANDARD C99

- funkcje inline
- deklaracje zmiennych w dowolnym miejscu w programie
- typ logiczny (`bool`), `long long int`
- tablice o zmiennej liczbie elementów
- komentarze w stylu C++ `//` to jest komentarz
- biblioteki, np.: `complex.h`, `stdbool.h`

Uwaga: nie wszystkie kompilatory wspierają pełny standard C99
dlatego ANSI C daje największą szansę na przenośność.

- każdą zmienną trzeba zadeklarować (określić typ i nazwę)
- przejrzystość: czytelne nazwy zmiennych i wcięcia
- nie zapomnij o średniku na końcu instrukcji
- operator przypisania `a=b` a operator porównania `a==b`
- najpierw deklaracja potem użycie
- inicjuj zmienne

 David Griffiths, Dawn Griffiths „*Rusz głową! C.*”, Helion, Gliwice, 2013.

 „Kurs programowania w C”, WikiBooks,
<http://pl.wikibooks.org/wiki/C>