Programowanie w języku C (EAR1S02005)

Politechnika Białostocka - Wydział Elektryczny

Automatyka i Robotyka, semestr II, studia stacjonarne I stopnia

Rok akademicki 2020/2021

Zajęcia nr 2 (24.02.2021)

dr inż. Jarosław Forenc

Język C - identyfikatory (nazwy)

- Dozwolone znaki: A-Z, a-z, 0-9, _ (podkreślenie)
- Długość nie jest ograniczona (rozróżnialne są 63 pierwsze znaki)
- Poprawne identyfikatory:

```
temp   u2   u_2   pole_kola   alfa   Beta   XyZ
```

- Pierwszym znakiem identyfikatora nie może być cyfra
- Nazwa zmiennej powinna być związana z jej zawartością

```
tempc   Tempc   TempC   TEMPC   TeMpC
```

- Powyższe zapisy oznaczają inne identyfikatory
- Jako nazw zmiennych nie można stosować słów kluczowych języka C

Język C - słowa kluczowe języka C

- W standardzie C11 zdefiniowane są 43 słowa kluczowe

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


Język C - Typy danych

Nazwa	Rozmiar (bajty)	Zakres wartości
char	1	-128 ... 127
int	4	-2147483648 ... 2147483647
float	4	$-3{,}4\cdot10^{38}$ ... $3{,}4\cdot10^{38}$
double	8	$-1{,}7\cdot10^{308}$ ... $1{,}7\cdot10^{308}$
void	–	–

- Słowa kluczowe wpływające na typy:
  - signed - liczba ze znakiem (dla typów char i int), np. signed char
  - unsigned - liczba bez znaku (dla typów char i int), np. unsigned int
  - short, long, long long - liczba krótka/długa (dla typu int), np. short int
  - long - większa precyzja (dla typu double), long double
- Zmienne typów int i long double mogą zajmować różną liczbę bajtów zależnie od środowiska programistycznego (kompilatora)

Język C - Stałe liczbowe

- Liczby całkowite (ang. integer) domyślnie mają typ int
  - system dziesiętny - domyślny, np. 1, 10, -250, 123456
  - system ósemkowy - 0 na początku, np. 011, 024
  - system szesnastkowy - 0x na początku, np. 0x2F, 0xab
- Przyrostki na końcu liczby całkowitej zmieniają typ
  - l lub L - typ long int, np. 10l, 10L
  - ll lub LL - typ long long int, np. 10ll, 10LL
  - u lub U - typ unsigned, np. 10u, 10U
- Liczby rzeczywiste domyślnie mają typ double
  - -2.41e+15, 14.15, 2e-5, 10.
- Przyrostki na końcu liczby rzeczywistej zmieniają typ:
  - l lub L - typ long double, np. 2.5L, 1.24e7l
  - f lub F - typ float, np. 3.14f, 1.24e7F

Język C - Stałe symboliczne (#define)

#define nazwa_stałej wartość_stałej

```
#include <stdio.h>
#define PI 3.14

int main(void)
{
    double pole, r = 1.5;

    pole = PI * r * r;
    printf("Pole = %g\n", pole);
    return 0;
}
```

```
Pole = 7.065
```

- Nazwy wyrażeń stałych zazwyczaj pisze się wielkimi literami
- W miejscu występowania stałej wstawiana jest jej wartość (przed właściwą kompilacją programu)

Język C - Operatory

Typ	Symbol
Arytmetyczne	`+ - * / %`
Inkrementacji / dekrementacji	`++ --`
Porównania (relacyjne)	`< > <= >= == !=`
Logiczne	`&& || !`
Bitowe	`& | ^ << >> ~`
Przypisania	`= += -= *= /= %= <<= >>= &= |= ^=`
Inne	`() [] & * -> . , ? : sizeof (typ)`

Język C - Priorytet operatorów (1/2)

Priorytet	Operator / opis
1	`++ --` (przyrostki) `() [] . ->`
2	`++ --` (przedrostki) `sizeof (typ)` `+ - ! ~ * &` (jednoargumentowe)
3	`* / %`
4	`+ -` (dwuargumentowe)
5	`<< >>`
6	`< > <= >=`
7	`== !=`
8	`&` (bitowy)
9	`^`

Język C - Priorytet operatorów (2/2)

Priorytet	Operator / opis
10	|
11	&&
12	||
13	? :
14	= += -= *= /= %= <<= >>= &= |= ^=
15	, (przecinek)

Język C - Wyrażenia arytmetyczne

- Kolejność wykonywania operacji wynika z priorytetu operatorów

```
w = a + b;
```
+ → =

```
w = a + b * c;
```
* → + → =

```
w = (a + b) * c;
```
(+) → * → =

```
w = (a + b) * (c + d);
```
(+) lub (+) → * → =

```
w = a + b + c;   →   w = ((a + b) + c);
w = x = y = a + b;   →   w = (x = (y = (a + b)));
```


Język C - Wyrażenia arytmetyczne

- Podczas dzielenia liczb całkowitych odrzucana jest część ułamkowa

$$w = \frac{5}{4}$$

```
5 / 4 = 1
```
```
5.0 / 4 = 1.25
```
```
5 / 4.0 = 1.25
```
```
5.0 / 4.0 = 1.25
```
```
5.0f / 4 = 1.25
```
```
5. / 4 = 1.25
```
```
(float) 5 / 4 = 1.25
```

Rzutowanie: (typ)

Język C - Funkcje matematyczne (math.h)

- Plik nagłówkowy math.h zawiera definicje wybranych stałych

Nazwa	Wartość	Znaczenie
M_PI	3.14159265358979323846	liczba pi
M_E	2.71828182845904523536	*e* - liczba Eulera
M_LN2	0.69314718055994530941 7	ln 2
M_SQRT2	1.41421356237309504880	$\sqrt{2}$

- W środowisku Microsoft Visual Studio użycie stałych wymaga definicji odpowiedniej stałej (przed #include <math.h>)

```
#define _USE_MATH_DEFINES
#include <math.h>
```

Język C - Funkcje matematyczne (math.h)

- Wybrane funkcje matematyczne:

Nazwa	Nagłówek	Znaczenie
`abs`	**int abs(int x);**	moduł x (x - całkowite)
`fabs`	**double fabs(double x);**	moduł x (x - rzeczywiste)
`sqrt`	**double sqrt(double x);**	pierwiastek kwadratowy x
`pow`	**double pow(double x, double y);**	x^y - x do potęgi y
`sin`	**double sin(double x);**	sinus argumentu x w radianach
`atan`	**double atan(double x);**	arcus tangens argumentu x
`atan2`	**double atan2(double y, double x);**	arcus tangens ilorazu y/x

- Większość funkcji ma po trzy wersje - dla argumentów typu: float, double i long double

Język C - Funkcja printf

- Ogólna składnia funkcji printf

```
printf("łańcuch_sterujący",arg1,arg2,...);
```

- W najprostszej postaci printf wyświetla tylko tekst

```
printf("Witaj swiecie");
```

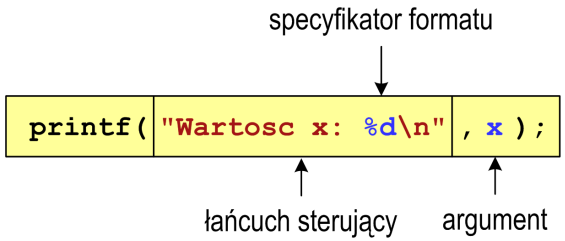
```
Witaj swiecie
```

- Do wyświetlenia wartości zmiennych konieczne jest zastosowanie specyfikatorów formatu, określających typ oraz sposób wyświetlania argumentów

```
%[znacznik][szerokość][.precyzja][modyfikator]typ
```


Język C - Funkcja printf

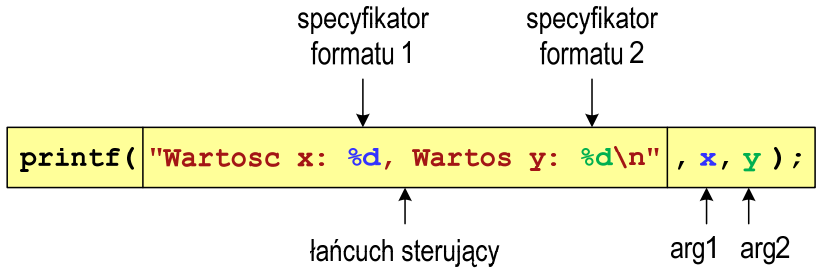
```
int x = 10;
printf("Wartosc x: %d\n", x);
```




```
Wartosc x: 10
```


Język C - Funkcja printf

```
int x = 10, y = 20;
printf("Wartosc x: %d, Wartosc y: %d\n", x, y);
```




```
Wartosc x: 10, Wartosc y: 20
```

Język C - Specyfikatory formatu (printf)

Typ w C	Specyfikator	Uwagi
char	%c	pojedynczy znak
	%d	kod ASCII znaku, liczba całkowita
char *	%s	łańcuch znaków, napis
int	%d %i	liczba całkowita, dziesiętna
	%o %O	liczba całkowita, ósemkowa
	%x %X	liczba całkowita, szesnastkowa
float double	%f	liczba rzeczywista
	%e %E	liczba rzeczywista, format naukowy
	%g %G	liczba rzeczywista (%f lub %e)

Język C - Funkcja printf

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%d], y = [%f]\n", x, y);
```

```
x = [123], y = [1.123457]
```

```
printf("x = [], y = []\n", x, y);
```

```
x = [], y = []
```

```
printf("x = [%d], y = [%d]\n", x, y);
```

```
x = [123], y = [-536870912]
```


Język C - Funkcja printf

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%6d], y = [%12f]\n", x, y);
```

```
x = [   123], y = [    1.123457]
```

```
printf("x = [%6d], y = [%12.3f]\n", x, y);
```

```
x = [   123], y = [       1.123]
```

```
printf("x = [%6d], y = [%.3f]\n", x, y);
```

```
x = [   123], y = [1.123]
```


Język C - Funkcja scanf

- Ogólna składnia funkcji scanf

```
scanf("specyfikatory",adresy_argumentów);
```

- Składnia specyfikatora formatu

```
%[szerokość][modyfikator]typ
```

- Argumenty są adresami obszarów pamięci, dlatego muszą być poprzedzone znakiem &

```
int x;
scanf("%d", &x);
```

Język C - Funkcja scanf

- Specyfikatory formatu w większości przypadków są takie same jak w przypadku funkcji printf
- Największa różnica dotyczy typów float i double

Typ w C	Specyfikator	Uwagi
float	%f	liczba rzeczywista
	%e %E	liczba rzeczywista, format naukowy
	%g %G	liczba rzeczywista (%f lub %e)
double	%lf	liczba rzeczywista
	%le %lE	liczba rzeczywista, format naukowy
	%lg %lG	liczba rzeczywista (%f lub %e)

Język C - Funkcja scanf

```
int a, b, c;
scanf("%d %d %d", &a, &b, &c);
```

- Wczytywane argumenty mogą być oddzielone od siebie dowolną liczbą białych (niedrukowalnych) znaków: spacja, tabulacja, enter

```
15 20 -30
```

```
15 20 -30<enter>
```

```
   15   20    -30
```

```
   15   20    -30<enter>
```

```
15
20
-30
```

```
15<enter>
20<enter>
-30<enter>
```