Przypomnienie

- char
  - typ liczbowy 1-bajtowy

Przypomnienie

- char
  - typ liczbowy 1-bajtowy
  - typ do przechowywania pojedynczego znaku tekstu
    - `char c = ‘a’`

Przypomnienie

- char
  - typ liczbowy 1-bajtowy
  - typ do przechowywania pojedynczego znaku tekstu
    - `char c = ‘a’`
    - `Char d = ‘\60’`
  - Znaki tekstu ↔ liczby (kod ASCII)

Character	ASCII
a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109

Character	ASCII
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Character	ASCII
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77

Character	ASCII
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

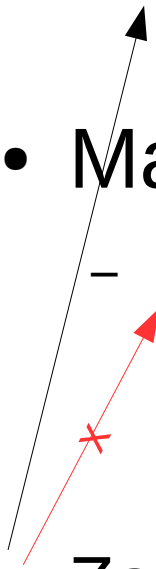
Character	ASCII
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Łańcuchy znaków

- Łańcuch znaków w stylu C
  = tablica znaków (char) zakończona 0
- Przykład (roboczy):
  - `char tekst[6] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, 0};`
    `puts(tekst);`
- Mamy do dyspozycji specjalną notację:
  - `char *tekst = „hello”;`
    `puts(tekst);`
- Zadanie:
  - Przed wypisaniem, zmień drugą literę na `‘a’`

Łańcuchy znaków

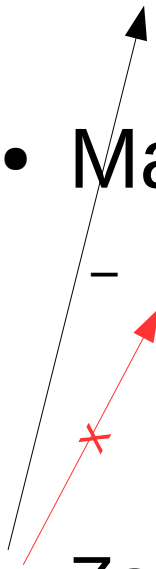
- Łańcuch znaków w stylu C
  = tablica znaków (char) zakończona 0
- Przykład (roboczy):
  - `char tekst[6] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, 0};`
    `puts(tekst);`
- Mamy do dyspozycji specjalną notację:
  - `char *tekst = „hello”;`
    `puts(tekst);`



- Zadanie:
  - Przed wypisaniem, zmień drugą literę na `‘a’`

Łańcuchy znaków

- Łańcuch znaków w stylu C
  = tablica znaków (char) zakończona 0
- Przykład (roboczy):
  - `char tekst[6] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, 0};`
    `puts(tekst);`
- Mamy do dyspozycji specjalną notację:
  - **`const`** `char *tekst = „hello”;`
    `puts(tekst);`



- Zadanie:
  - Przed wypisaniem, zmień drugą literę na `‘a’`

Wiersz polecenia

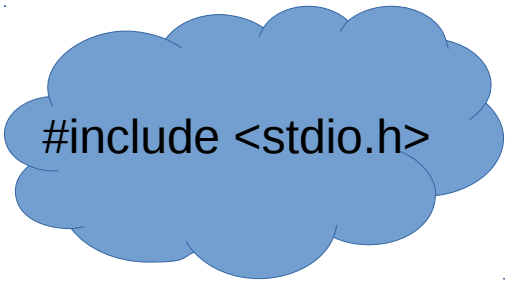
- Czytanie z wiersza poleceń
  (z parametrów podanych przy uruchamianiu)
  - `int main(int argc, char **argv)`
    - `argc`: liczba argumentów
    - `argv[0]`: ścieżka do naszego programu
    - `argv[1]`, `argv[2]` ... `argv[argc-1]`:
      kolejne argumenty (tekstowe)

Wiersz polecenia

- Czytanie z wiersza poleceń (z parametrów podanych przy uruchamianiu)
  - `int main(int argc, char **argv)`
    - `argc`: liczba argumentów
    - `argv[0]`: ścieżka do naszego programu
    - `argv[1]`, `argv[2]` ... `argv[argc-1]`: kolejne argumenty (tekstowe)
- Konwersja na inne typy:
  - `int atoi(const char *nptr);`
  - `long atol(const char *nptr);`
  - `long long atoll(const char *nptr);`
  - `double atof(const char *nptr);`

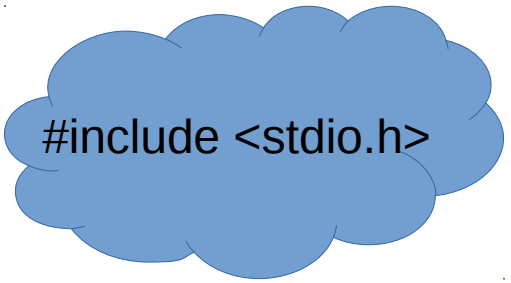
Wypisywanie na standardowe wyjście

- Pojedynczy znak tekstu
  - `int putchar(int c);`
    - `putchar('x');`



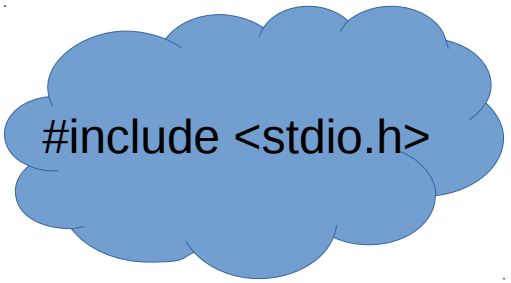
Wypisywanie na standardowe wyjście

- Pojedynczy znak tekstu
  - `int putchar(int c);`
    - `putchar('x');`
- Łańcuch znaków
  - `int puts(const char *s);` // dodaje znak nowej linii
    - `puts("Hello World!");`



Wypisywanie na standardowe wyjście

- Pojedynczy znak tekstu
  - `int putchar(int c);`
    - `putchar('x');`
- Łańcuch znaków
  - `int puts(const char *s);` // dodaje znak nowej linii
    - `puts("Hello World!");`
- Dowolne wartości w zadanym formacie
  - `int printf(const char *format, ...);`
    - `printf("%d+%d=%d", x, y, x+y);`



Format printf

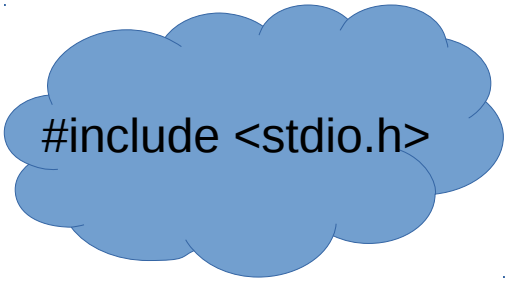
Data type		Format specifier
Integer	short signed	%d or %I
	short unsigned	%u
	long singed	%ld
	long unsigned	%lu
	unsigned hexadecimal	%x
	unsigned octal	%o
Real	float	%f
	double	%lf
Character	signed character	%c
	unsigned character	%c
String		%s

Format printf

- Dodatkowo:
  - Minimalna szerokość pola:
    - %20s – tekst poprzedzony spacjami do 20 znaków
    - %6d – liczba (int) poprzedzona spacjami do 6 znaków
    - %**0**6d – liczba (int) poprzedzona **zerami** do 6 znaków
  - Liczba miejsc po przecinku:
    - %8.3f – liczba (float) o szerokości 8 znaków z **3** miejscami po przecinku
  - Wyświetlanie znaku plus dla liczb dodatnich:
    - %+7d – liczba (int) zawierająca znak plus (o ile dodatnia) poprzedzona spacjami do 7 znaków

Czytanie ze standardowego wejścia

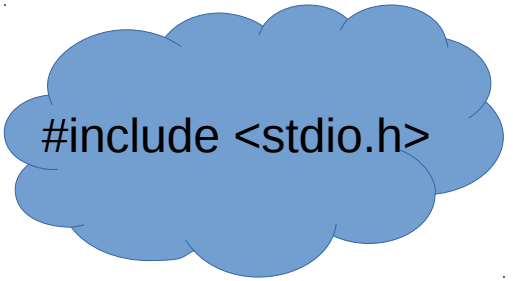
- Pojedynczy znak
  - `int getchar(void);`
    - UWAGA! Może zwrócić wartość EOF



Czytanie ze standardowego wejścia

- Pojedynczy znak
  - `int getchar(void);`
    - UWAGA! Może zwrócić wartość EOF
- Dowolne wartości w zadanym formacie
  - `int scanf(const char *format, ...);`
    - format jest analogiczny do printf
    - wartości podajemy **jako wskaźniki**

Czytanie ze standardowego wejścia



- Pojedynczy znak
  - `int getchar(void);`
    - UWAGA! Może zwrócić wartość EOF
- Dowolne wartości w zadanym formacie
  - `int scanf(const char *format, ...);`
    - format jest analogiczny do printf
    - wartości podajemy **jako wskaźniki**
    - `int a;`
      `scanf(„%d”, &a);` // wczyta jedną liczbą do zmiennej a

Zadanie

- Napisz program, który pobierze **z linii poleceń** wszystkie podane wartości (jako int),
  a następnie obliczy (i wypisze) ich sumę.

  (2 minuty)

Zadanie

- Napisz program, który pobierze **ze standardowego wejścia** współczynniki a, b, c (double lub float) równania kwadratowego

$$ax^2 + bx + c = 0$$

a następnie znajdzie (i wypisze) rozwiązania tego równania.

(5 minut)

Zadanie

- Napisz program, który pobierze **ze standardowego wejścia** współczynniki a, b, c (double lub float) równania kwadratowego

$$ax^2 + bx + c = 0$$

a następnie znajdzie (i wypisze) rozwiązania tego równania.

(OK, 10 minut)

$\sqrt{x}$

#include <math.h>

```
double sqrt(double x);
float sqrtf(float x);
long double sqrtl(long double x);
```

Struktury

- Struktura = własny typ, złożony z kilku **nazwanych** pól
  - ```
    struct MyOrder {
      int quantity;
      double unitPrice;
    };
    ```
- Użycie:
  - ```
    struct MyOrder order;
    order.quantity = 100;
    order.unitPrice = 5.99;
    ```

Struktury

- Struktura = własny typ, złożony z kilku **nazwanych** pól
  - ```
    typedef struct {
      int quantity;
      double unitPrice;
    } MyOrder;
    ```
- Użycie:
  - ```
    MyOrder order;
    order.quantity = 100;
    order.unitPrice = 5.99;
    ```

Struktury & tablice

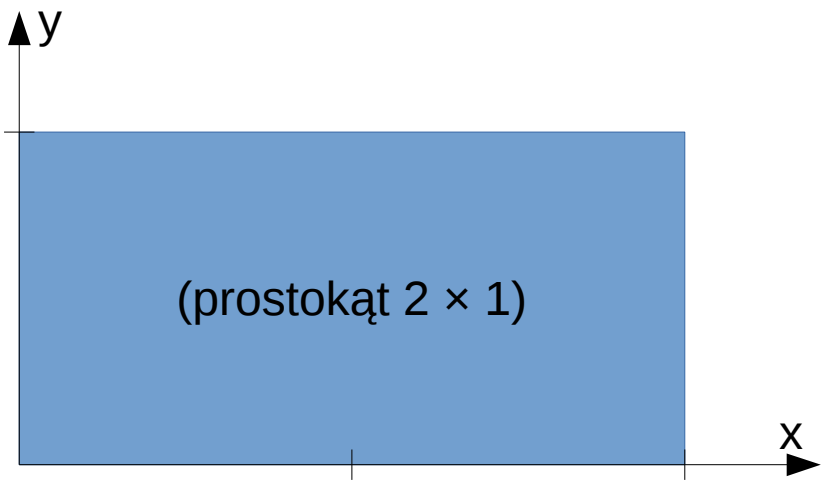
- Tablica może składać się ze struktur!
  - ```
    struct MyOrder {
      int quantity;
      double unitPrice;
    };
    struct MyOrder myOrders[10];
    ...
    myOrders[3].quantity = 1;
    myOrders[3].unitPrice = 2.59;
    ...
    myOrders[5] = myOrders[3];
    ```
    ```

Struktury & tablice

- Struktura może składać się z tablic!
  - ```
    struct Rectangle {
      double x[2], y[2];
    };
    struct Rectangle rect;
    rect.x[0] = 0.0;
    rect.x[1] = 2.0;
    rect.y[0] = 0.0;
    rect.y[1] = 1.0;
    ```
    ```

Struktury & tablice

- Struktura może składać się z tablic!
  - ```
    struct Rectangle {
      double x[2], y[2];
    };
    struct Rectangle rect;
    rect.x[0] = 0.0;
    rect.x[1] = 2.0;
    rect.y[0] = 0.0;
    rect.y[1] = 1.0;
    ```


    ```

Zadanie

- Napisz program, który pobierze **ze standardowego wejścia** współczynniki a, b, c (double lub float) równania kwadratowego

$$ax^2 + bx + c = 0$$

a następnie znajdzie (i wypisze) rozwiązania tego równania.

Funkcja obliczająca rozwiązanie ma mieć następującą deklarację:

```
int solve(struct Equation eq, double* x)
```

gdzie

- Rozwiązania zostaną zapisane do tablicy x
- Liczba rozwiązań zostanie zwrócona z funkcji
- ```
  struct Equation {
    double a, b, c;
  }
  ```
  ```

Słowa kluczowe (keywords) języka C

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

Słowa kluczowe (keywords) języka C

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

Słowa kluczowe (keywords) języka C

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