Informatyka 1 (ES1F1002)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia stacjonarne I stopnia

Rok akademicki 2022/2023

Wykład nr 2 (10.10.2022)

dr inż. Jarosław Forenc

Plan wykładu nr 2

- Język C
  - deklaracje zmiennych i stałych
  - operatory, priorytet operatorów
  - wyrażenia, instrukcje
  - wyrażenia arytmetyczne, funkcje matematyczne (math.h)
  - funkcje printf i scanf
- Systemy liczbowe
  - liczby i cyfry
  - systemy pozycyjne i niepozycyjne
  - konwersje między systemami liczbowymi

Przykład: zamiana wzrostu w cm na stopy i cale

```
#include <stdio.h>

int main(void)
{
    float cm;     /* wzrost w cm */
    float stopy;  /* wzrost w stopach */
    float cale;   /* wzrost w calach */

    printf("Podaj wzrost w cm: ");
    scanf("%f",&cm);

    stopy = cm / 30.48f;
    cale = cm / 2.54f;

    printf("%f [cm] = %f [ft]\n",cm,stopy);
    printf("%f [cm] = %f [in]\n",cm,cale);

    return 0;
}
```

```
Podaj wzrost w cm: 175
175.000000 [cm] = 5.741470 [ft]
175.000000 [cm] = 68.897636 [in]
```


Język C - Deklaracje zmiennych i stałych

- Zmienne (ang. variables) - zmieniają swoje wartości podczas pracy programu
- Stałe (ang. constants) - mają wartości ustalane przed uruchomieniem programu i pozostają niezmienione przez cały czas jego działania
- Deklaracja nadaje zmiennej / stałej nazwę, określa typ przechowywanej wartości i rezerwuje odpowiednio obszar pamięci
- Deklaracje zmiennych:

```
int x;
float a, b;
char zn1;
```

- Deklaracje stałych:

```
const int y = 5;
const float c = 1.25f;
const char zn2 = 'Q';
```

- Inicjalizacja zmiennej:

```
int x = -10;
```

Język C - Stałe symboliczne (#define)

- Dyrektywa preprocesora #define umożliwia definiowanie tzw. stałych symbolicznych

#define nazwa_stałej wartość_stałej

```
#define PI 3.14
#define KOMUNIKAT "Zaczynamy!!!\n"
```

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

Przykład: pole i obwód koła

```
#include <stdio.h>
#define PI 3.14
#define KOMUNIKAT "Zaczynamy!!!\n"

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

    printf(KOMUNIKAT);
    pole = PI * r * r;
    obwod = 2 * PI * r;

    printf("Pole = %g\n", pole);
    printf("Obwod = %g\n", obwod);

    return 0;
}
```


Przykład: pole i obwód koła

```
/***
...
#endif  /* _INC_STDIO */

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

    printf("Zaczynamy!!!\n");
    pole = 3.14 * r * r;
    obwod = 2 * 3.14 * r;

    printf("Pole = %g\n", pole);
    printf("Obwod = %g\n", obwod);

    return 0;
}
```

zawartość pliku stdio.h

```
Zaczynamy!!!
Pole = 7.065
Obwod = 9.42
```


Język C - Operatory

- Operator - symbol lub nazwa operacji
- Argumenty operatora nazywane są operandami
- Operator jednoargumentowy

- Operator dwuargumentowy

- Operator trójargumentowy

- Operator wieloargumentowy

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

- Wyrażenie (ang. expression) - kombinacja operatorów i operandów

```
4    -6    4+2.1    x=5+2    a>3    x>5&&x<8
```

- Każde wyrażenie ma typ i wartość

Wyrażenie	Typ	Wartość
4	int	4
-6	int	-6
4 + 2.1	double	6.1
x = 5 + 2	*typ x*	7
a > 3	int	1 (prawda) / 0 (fałsz)
x > 5 && x < 8	int	1 (prawda) / 0 (fałsz)

Język C - instrukcje

- Instrukcja (ang. statement) - główny element, z którego zbudowany jest program, kończy się średnikiem

Wyrażenie: `x = 5` Instrukcja: `x = 5;`

- Język C za instrukcję uznaje każde wyrażenie, na którego końcu znajduje się średnik

```
8;
x;
3 + 4;
a > 5;
```

- Powyższe instrukcje są poprawne, ale nie dają żadnego efektu

Język C - instrukcje

- Podział instrukcji:
  - proste - kończą się średnikiem
  - złożone - kilka instrukcji zawartych pomiędzy nawiasami klamrowymi
- Typy instrukcji prostych:
  - deklaracji: `int x;`
  - przypisania: `x = 5;`
  - wywołania funkcji: `printf("Witaj swiecie\n");`
  - strukturalna: `while(x > 0) x--;`
  - pusta: `;`

Język C - wyrażenia arytmetyczne

- Wyrażenia arytmetyczne mogą zawierać:
  - stałe liczbowe, zmienne, stałe
  - operatory: `+ - * / % = ( )` i inne
  - wywołania funkcji (plik nagłówkowy math.h)
- Kolejność wykonywania operacji wynika z priorytetu operatorów

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

Język C - wyrażenia arytmetyczne

- Kolejność wykonywania operacji

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

- Zapis wyrażeń arytmetycznych

$w = \frac{a+b}{c+d}$

```
w = a + b / c + d;
```
ŹLE

```
w = (a + b) / (c + d);
```
DOBRZE

$w = \frac{a+b}{c \cdot d}$

```
w = (a + b) / c * d;
```
ŹLE

```
w = (a + b) / (c * d);
```
DOBRZE

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 Visual Studio 2008 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

- Wszystkie funkcje mają po trzy wersje - dla argumentów typu: float, double i long double

Przykład: częstotliwość rezonansowa

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

int main(void)
{
    double L, C, fr;

    printf("Podaj L [H]:  "); scanf("%lf",&L);
    printf("Podaj C [F]:  "); scanf("%lf",&C);

    fr = 1/(2*M_PI*sqrt(L*C));

    printf("--------------------\n");
    printf("fr [Hz]:  %.3f\n",fr);

    return 0;
}
```

```
Podaj L [H]:  0.01
Podaj C [F]:  1e-6
--------------------
fr [Hz]:  1591.549
```

$$f_r = \frac{1}{2\pi\sqrt{LC}}$$

Język C - Funkcja printf

- Ogólna składnia funkcji printf

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

- W najprostszej postaci printf wyświetla tylko tekst

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

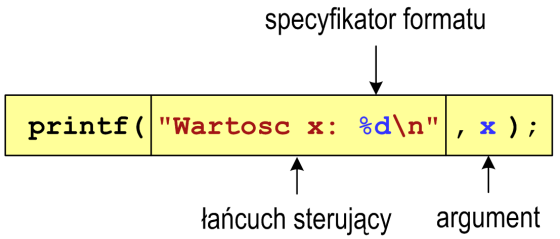
```
Witaj swiecie
```

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

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


Język C - Funkcja printf

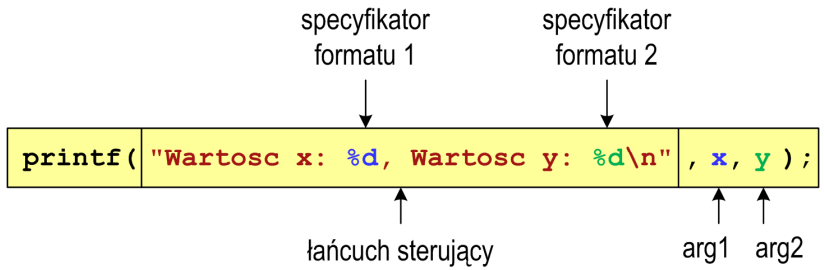
```
int x = 10;
printf("Wartosc x: %d\n", x);
```




```
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 = [1.234568]
```

```
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 = [    1.234568]
```

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

```
x = [   123], y = [       1.235]
```

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

```
x = [   123], y = [1.235]
```


Język C - Funkcja printf

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

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

```
x = [  +123], y = [   +1.234568]
```

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

```
x = [123   ], y = [1.234568    ]
```

```
printf("x = [%06d], y = [%012f]\n", x, y);
```

```
x = [000123], y = [00001.234568]
```


Język C - Funkcja printf

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

```
printf("x = [%d], y = [%f]\n", x+321, y*25.5f);
```

```
x = [444], y = [31.481482]
```

```
printf("x = [%d], y = [%f]\n", 123, 2.0f*sqrt(y));
```

```
x = [123], y = [2.222222]
```

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>
```


Liczby i cyfry

- Liczba - pojęcie abstrakcyjne, abstrakcyjny wynik obliczeń, wartość
  - umożliwia wyrażenie wyniku liczenia przedmiotów oraz mierzenia wielkości
- Cyfra - umowny znak (symbol) stosowany do zapisu liczby
  - liczba znaków służących do zapisu jest zależna od systemu liczbowego i przyjętego sposobu zapisu
  - system dziesiętny - 10 znaków
  - system szesnastkowy - 16 znaków
  - system rzymski - 7 znaków
- Cyfry rzymskie

I	V	X	L	C	D	M
1	*5*	*10*	*50*	*100*	*500*	*1000*

Liczby i cyfry

- Cyfry arabskie (pochodzą z Indii)
  - arabskie, standardowe europejskie

1	2	3	4	5	6	7	8	9	0

  - indyjsko-arabskie

١	٢	٣	٤	٥	٦	٧	٨	٩	٠
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*0*

  - wschodnio-indyjsko-arabskie

১	২	৩	৪	৫	৬	৭	৮	৯	০
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*0*

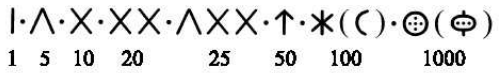
- W niektórych systemach jako cyfry stosowane są litery, np.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*

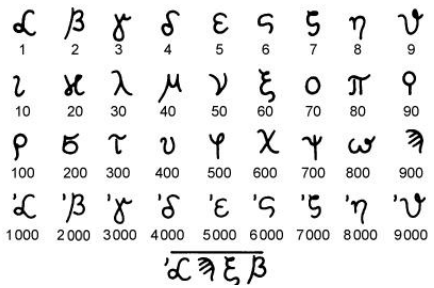
Liczby i cyfry

- Inne przykłady zapisu cyfr i liczb:

cyfry etruskie



cyfry grecko-jońskie

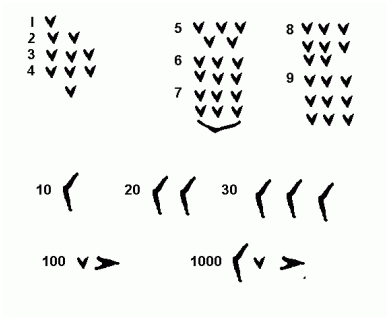


cyfry w pisowni chińskiej

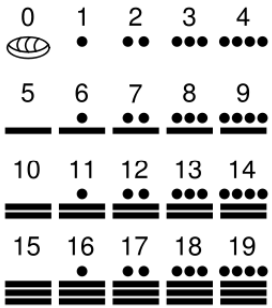
Liczby i cyfry

- Inne przykłady zapisu cyfr i liczb:

liczby w piśmie klinowym (Babilończycy)

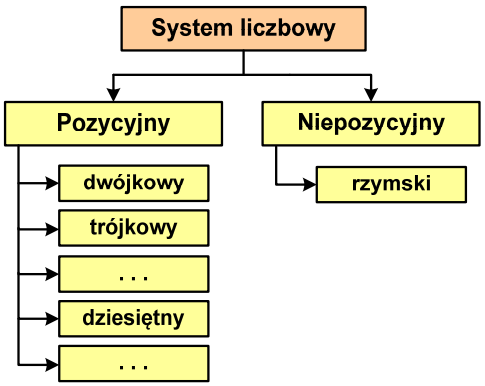


system prekolumbijski



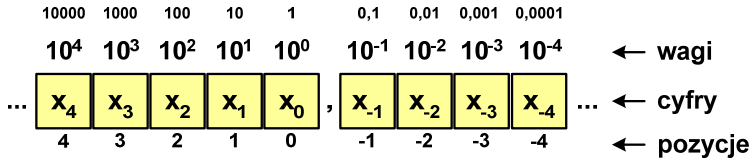
Systemy liczbowe

- System liczbowy - zbiór zasad umożliwiających zapis liczb za pomocą cyfr oraz wykonywanie działań na tych liczbach

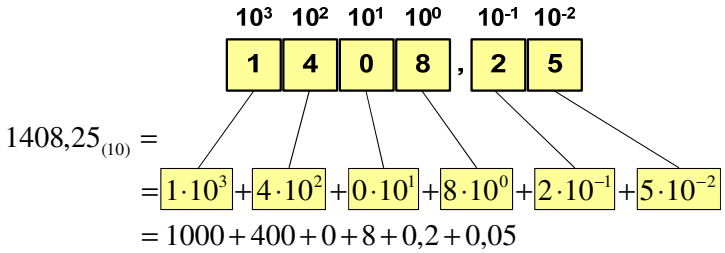


- Pozycyjny - znaczenie cyfry jest zależne od miejsca (pozycji), które zajmuje ona w liczbie
  - system dziesiętny - liczba 111 (każda cyfra ma inne znaczenie)
- Niepozycyjny - znaczenie cyfry jest niezależne od miejsca położenia w liczbie
  - system rzymski - liczba III

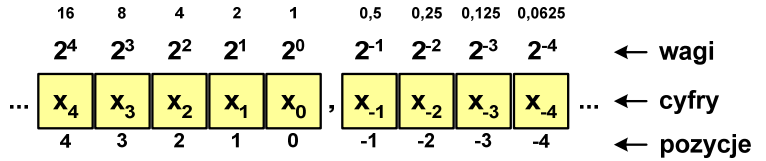
System dziesiętny (ang. decimal)



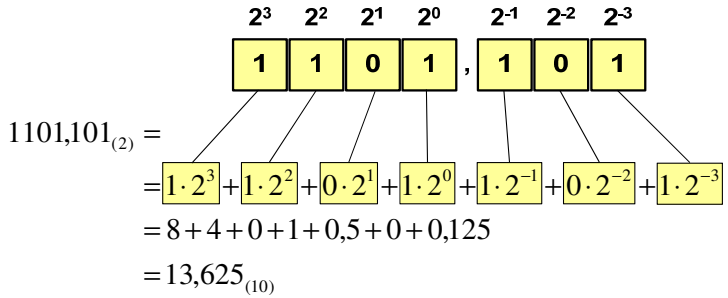
- p - podstawa systemu pozycyjnego, D - zbiór dozwolonych cyfr
- p = 10, D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}



System dwójkowy (ang. binary)

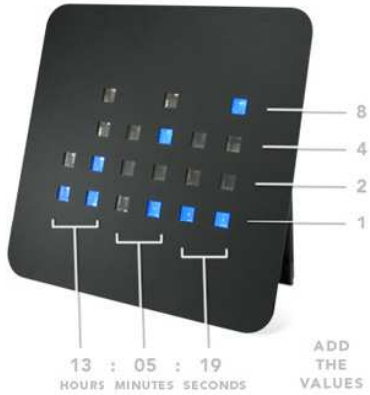


- w systemie dwójkowym: p = 2, D = {0, 1}



System dwójkowy - zastosowania

- Powszechnie używany w informatyce, technice cyfrowej



System szesnastkowy (ang. hexadecimal)

- System heksadecymalny
- p = 16, D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
- Powszechnie używany w informatyce - jeden bajt można zapisać za pomocą tylko dwóch cyfr szesnastkowych

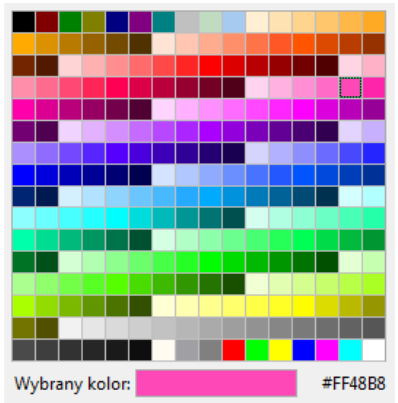
$$3A5D_{(16)} = 3\cdot16^3 + 10\cdot16^2 + 5\cdot16^1 + 13\cdot16^0 = 14941_{(10)}$$

- Sposoby zapisu liczb w systemie szesnastkowym:

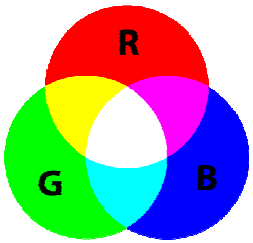
3A5Dh	0x3A5D	#3A5D
$3A5D_{(16)}$	$3A5D_{16}$	$3A5D_{hex}$
$(3A5D)_{hex}$	$(3A5D)_{16}$	$3A5D

System szesnastkowy - zastosowania

- Zapis 24-bitowego koloru RGB (Red-Green-Blue), 16 mln kolorów
- Każda barwa przyjmuje wartość z zakresu: $0..255_{(10)}$, $00..FF_{(16)}$



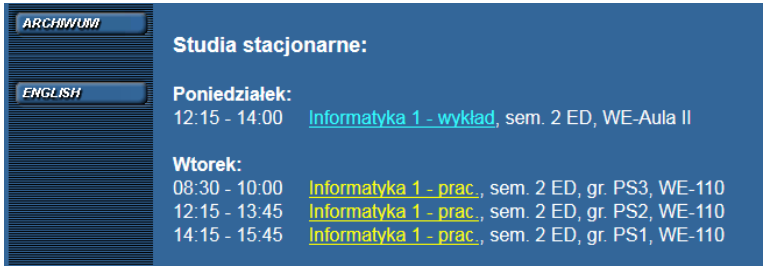
#FF48B8



System szesnastkowy - zastosowania

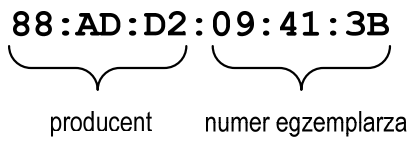
- Zapis 24-bitowego koloru RGB (Red-Green-Blue), 16 mln kolorów
- Kolory w dokumentach HTML:

```
<BODY bgcolor="#336699" text="#000000" link="#FFFF00"
 vlink="#33FFFF" alink="#FF0000">
```



System szesnastkowy - zastosowania

- 48-bitowy adres fizyczny urządzenia (MAC - Media Access Control)



- http://hwaddress.com

Przykład systemu niepozycyjnego - system rzymski

- W systemie rzymskim posługujemy się siedmioma znakami:

  I - 1 V - 5 X - 10 L - 50 C - 100 D - 500 M - 1000
- Za pomocą dostępnych symboli można określić liczby od 1 do 3999
- System addytywny - wartość liczby określa się na podstawie sumy wartości cyfr, np.
  - II (1 + 1 = 2), XXX (10 + 10 + 10 = 30)
  - CLX (100 + 50 + 10 = 160), MMXII (1000 + 1000 + 10 + 1 + 1 = 2012)
- Wyjątkiem od powyższej zasady są liczby do opisu których używa się odejmowania, np.
  - IV (5 - 1 = 4), IX (10 - 1 = 9), XL (50 - 10 = 40), XC (100 - 10 = 90)
- Stosowany w łacińskiej części Europy do końca Średniowiecza
- Niewygodny w prowadzeniu nawet prostych działań arytmetycznych, brak ułamków

Przykład systemu niepozycyjnego - system rzymski

- Zasady tworzenia liczb:
  - zestawiamy odpowiednie znaki od oznaczającego liczbę największą do oznaczającego liczbę najmniejszą

    XVI = 10(X) + 5(V) + 1(I) = 16
  - jeżeli składnik liczby, którą piszemy, jest wielokrotnością liczby nominalnej, wtedy zapisywany jest z użyciem kilku następujących po sobie znaków

    CCC = 100(C) + 100(C) + 100(C) = 300
  - dodatkowo należy zachować zasadę nie pisania czterech tych samych znaków po sobie, lecz napisać jeden znak wraz ze znakiem oznaczającym wartość większą o jeden rząd liczbowy

    CD = 500(D) - 100(C) = 400

Przykład systemu niepozycyjnego - system rzymski

- Zasady odczytu liczb:
  - cyfry jednakowe są dodawane

    MMM = 1000(M) + 1000(M) + 1000(M) = 3000
  - cyfry mniejsze stojące przed większymi są odejmowane od nich

    CDXCIV = 500(D) - 100(C) + 100(C) - 10(X) + 5(V) - 1(I) = 494
  - cyfry mniejsze stojące za większymi są do nich dodawane

    MDCLX = 1000(M) + 500(D) + 100(C) + 50(L) + 10 (X) = 1660

Konwersja na system dziesiętny

- p = 4, D = {0, 1, 2, 3}

4^4	4^3	4^2	4^1	4^0
2	1	3	0	2

$$21302_{(4)} = ?_{(10)}$$
$$21302_{(4)} = 2\cdot 4^0 + 0\cdot 4^1 + 3\cdot 4^2 + 1\cdot 4^3 + 2\cdot 4^4$$
$$21302_{(4)} = 2\cdot 1 + 0\cdot 4 + 3\cdot 16 + 1\cdot 64 + 2\cdot 256$$
$$21302_{(4)} = 2 + 0 + 48 + 64 + 512 = 626_{(10)}$$

- p = 17, D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G}

17^3	17^2	17^1	17^0
A	C	2	4

$$AC24_{(17)} = ?_{(10)}$$
$$AC24_{(17)} = 4\cdot 17^0 + 2\cdot 17^1 + 12\cdot 17^2 + 10\cdot 17^3$$
$$AC24_{(17)} = 4\cdot 1 + 2\cdot 17 + 12\cdot 289 + 10\cdot 4913$$
$$AC24_{(17)} = 4 + 34 + 3468 + 49130 = 52636_{(10)}$$

Konwersja na system dziesiętny (schemat Hornera)

- p = 4, D = {0, 1, 2, 3}

$$21302_{(4)} = w_{(10)} \qquad x_4\, x_3\, x_2\, x_1\, x_0 = w_{(10)}$$
$$w_{(10)} = 0$$
$$w_{(10)} = x_4 + w_{(10)} \cdot p = 2 + 0\cdot 4 = 2$$
$$w_{(10)} = x_3 + w_{(10)} \cdot p = 1 + 2\cdot 4 = 9$$
$$w_{(10)} = x_2 + w_{(10)} \cdot p = 3 + 9\cdot 4 = 39$$
$$w_{(10)} = x_1 + w_{(10)} \cdot p = 0 + 39\cdot 4 = 156$$
$$w_{(10)} = x_0 + w_{(10)} \cdot p = 2 + 156\cdot 4 = 626_{(10)}$$

Konwersja z systemu dziesiętnego na dowolny

- zamiana liczby z systemu p = 10 na system p = 2

$626_{(10)} = ?_{(2)}$ $\qquad$ $626_{(10)} = 1001110010_{(2)}$

626 / 2	= 313	*reszta*	0
313 / 2	= 156	*reszta*	1
156 / 2	= 78	*reszta*	0
78 / 2	= 39	*reszta*	0
39 / 2	= 19	*reszta*	1
19 / 2	= 9	*reszta*	1
9 / 2	= 4	*reszta*	1
4 / 2	= 2	*reszta*	0
2 / 2	= 1	*reszta*	0
1 / 2	= 0	*reszta*	1

kolejność odczytywania cyfr liczby w systemie dwójkowym

kończymy, gdy liczba dziesiętna ma wartość 0

Konwersja z systemu dziesiętnego na dowolny

- zamiana liczby z systemu p = 10 na system p = 7

$626_{(10)} = ?_{(7)}$ $\qquad$ $626_{(10)} = 1553_{(7)}$

626 / 7	= 89	*reszta*	3
89 / 7	= 12	*reszta*	5
12 / 7	= 1	*reszta*	5
1 / 7	= 0	*reszta*	1

- zamiana liczby z systemu p = 10 na system p = 14

$626_{(10)} = ?_{(14)}$ $\qquad$ $626_{(10)} = 32A_{(14)}$

626 / 14	= 44	*reszta*	10 → A
44 / 14	= 3	*reszta*	2
3 / 14	= 0	*reszta*	3

Szybkie konwersje: 2 → 4, 8, 16 4, 8, 16 → 2

2 → 4

$110110011_{(2)} = ?_{(4)}$

$\underbrace{01}_{1} \mid \underbrace{10}_{2} \mid \underbrace{11}_{3} \mid \underbrace{00}_{0} \mid \underbrace{11}_{3}$

$110110011_{(2)} = 12303_{(4)}$

2 → 8

$10110011_{(2)} = ?_{(8)}$

$\underbrace{010}_{2} \mid \underbrace{110}_{6} \mid \underbrace{011}_{3}$

$10110011_{(2)} = 263_{(8)}$

2 → 16

$1011010_{(2)} = ?_{(16)}$

$\underbrace{0101}_{5} \mid \underbrace{1010}_{A}$

$1011010_{(2)} = 5A_{(16)}$

4 → 2

$12303_{(4)} = ?_{(2)}$

$\underbrace{01}_{1} \; \underbrace{10}_{2} \; \underbrace{11}_{3} \; \underbrace{00}_{0} \; \underbrace{11}_{3}$

$12303_{(4)} = 110110011_{(2)}$

8 → 2

$263_{(8)} = ?_{(2)}$

$\underbrace{010}_{2} \; \underbrace{110}_{6} \; \underbrace{011}_{3}$

$263_{(8)} = 10110011_{(2)}$

16 → 2

$5A_{(16)} = ?_{(2)}$

$\underbrace{0101}_{5} \; \underbrace{1010}_{A}$

$5A_{(16)} = 1011010_{(2)}$

Koniec wykładu nr 2

Dziękuję za uwagę!