



Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Materiały do wykładu z przedmiotu:
Informatyka
Kod: EDS1A1 007

WYKŁAD NR 2

Opracował: **dr inż. Jarosław Forenc**
Białystok 2018

Materiały zostały opracowane w ramach projektu „PB2020 - Zintegrowany Program Rozwoju Politechniki Białostockiej” realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

Plan wykładu nr 2

- Wyrażenia i instrukcje, wyrażenia arytmetyczne
- Funkcje printf i scanf
- Instrukcja warunkowa if, operator warunkowy
- Instrukcja switch
- Pętla for
- Operatory ++ i --

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;
```

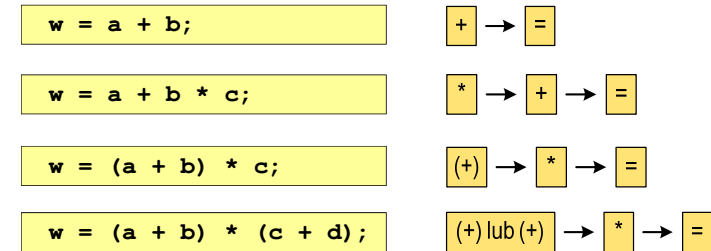
- Powyższe instrukcje są poprawne, ale nie dają żadnego efektu

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: `;`

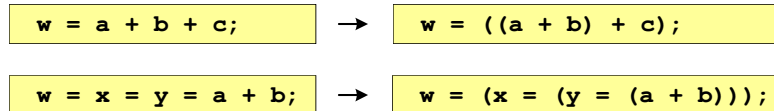
Język C - Wyrażenia arytmetyczne

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

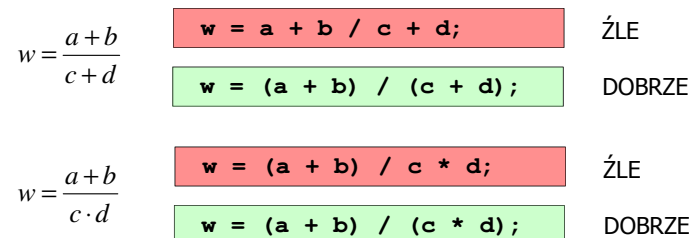


Język C - Wyrażenia arytmetyczne

- Kolejność wykonywania operacji

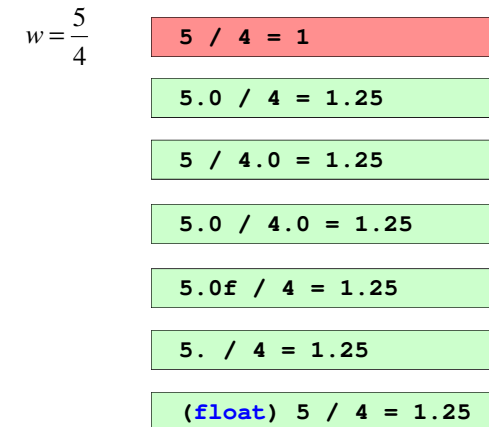


- Zapis wyrażen arytmetycznych



Język C - Wyrażenia arytmetyczne

- Podczas dzielenia liczb całkowitych odrzucana jest część ułamkowa



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	0.693147180559945309417	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	<code>int abs(int x);</code>	moduł x (x - całkowite)
fabs	<code>double fabs(double x);</code>	moduł x (x - rzeczywiste)
sqrt	<code>double sqrt(double x);</code>	piersiastek kwadratowy x
pow	<code>double pow(double x, double y);</code>	x^y - x do potęgi y
sin	<code>double sin(double x);</code>	sinus argumentu x w radianach
atan	<code>double atan(double x);</code>	arcus tangens argumentu x
atan2	<code>double atan2(double y, double x);</code>	arcus tangens ilorazu y/x

- Wszystkie funkcje mają 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");
```

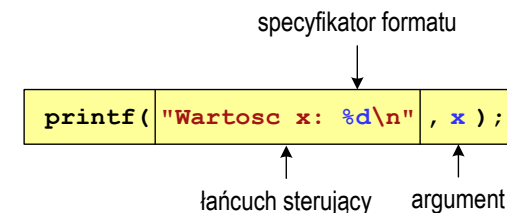
Witaj swiecie

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

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

Język C - Funkcja printf

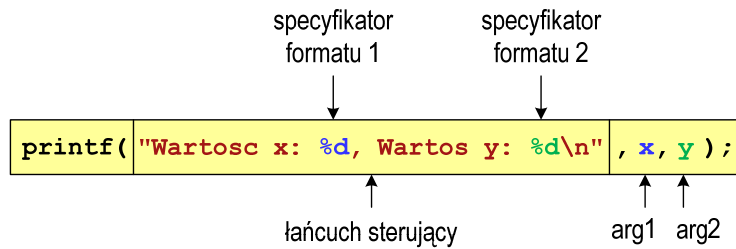
```
int x = 10;  
printf("Wartosc x: %d\n", x);
```



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

```
x = [123], y = [1.123457]
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);
printf("x = [%6d], y = [%12.3f]\n", x, y);
printf("x = [%6d], y = [%.3f]\n", x, y);
```

```
x = [ 123], y = [ 1.123457]
x = [ 123], y = [ 1.123]
x = [ 123], y = [1.123]
```

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

Język C - Funkcja printf

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

```
printf("x = [%+6d], y = [%+12f]\n", x, y);  
printf("x = [%-6d], y = [%-12f]\n", x, y);  
printf("x = [%06d], y = [%012f]\n", x, y);
```

```
x = [ +123], y = [ +1.123457]  
x = [123 ], y = [1.123457 ]  
x = [000123], y = [00001.123457]
```

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

Język C - Funkcja printf

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

```
printf("x = [%d], y = [%f]\n", x, y);  
printf("x = [%d], y = [%f]\n", x+321, y*25.5f);  
printf("x = [%d], y = [%f]\n", 123, 2.0f*sqrt(y));
```

```
x = [123], y = [1.123457]  
x = [444], y = [28.648149]  
x = [123], y = [2.119865]
```

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>
```

Język C - Pierwiastek kwadratowy

```
#include <stdio.h>  
#include <math.h>  
  
int main(void)  
{  
    float x, y;  
  
    printf("Podaj liczbe: ");  
    scanf("%f", &x);  
  
    y = sqrt(x);  
  
    printf("Pierwiastek liczby: %f\n", y);  
  
    return 0;  
}
```

```
Podaj liczbe: 15  
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15  
Pierwiastek liczby: -1.#IND00
```

Język C - Pierwiastek kwadratowy

```
#include <stdio.h>  
#include <math.h>  
  
int main(void)  
{  
    float x, y;  
  
    printf("Podaj liczbe: ");  
    scanf("%f", &x);  
  
    if (x>=0)  
    {  
        y = sqrt(x);  
        printf("Pierwiastek liczby: %f\n", y);  
    }  
    else  
        printf("Blad! Liczba ujemna\n");  
  
    return 0;  
}
```

```
Podaj liczbe: 15  
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15  
Blad! Liczba ujemna
```

Język C - instrukcja warunkowa if

```
if (wyrażenie)  
    instrukcja1
```

- jeśli **wyrażenie** jest prawdziwe, to wykonywana jest **instrukcja1**
- gdy **wyrażenie** jest fałszywe, to **instrukcja1** nie jest wykonywana

```
if (wyrażenie)  
    instrukcja1  
else  
    instrukcja2
```

- jeśli **wyrażenie** jest prawdziwe, to wykonywana jest **instrukcja1**, zaś **instrukcja2** nie jest wykonywana
- gdy **wyrażenie** jest fałszywe, to wykonywana jest **instrukcja2**, zaś **instrukcja1** nie jest wykonywana

- Wyrażenie w nawiasach:

- prawdziwe** - gdy jego wartość jest różna od zera
- fałszywe** - gdy jego wartość jest równa zero

Język C - instrukcja warunkowa if

```
if (wyrażenie)  
instrukcja
```

■ Instrukcja:

- **prosta** - jedna instrukcja zakończona średnikiem
- **złożona** - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
if (x>0)  
    printf("inst1");
```

```
if (x>0)  
{  
    printf("inst1");  
    printf("inst2");  
    ...  
}
```

Język C - instrukcja warunkowa if

```
if (wyr)  
    instr;
```

```
if (wyr)  
    instr;  
else  
    instr;
```

```
if (wyr)  
{  
    instr;  
    instr;  
}  
else  
    instr;
```

```
if (wyr)  
{  
    instr;  
}  
else  
{  
    instr;  
}
```

```
if (wyr)  
{  
    instr;  
    instr;  
}
```

```
if (wyr)  
{  
    instr;  
    instr;  
}  
else  
{  
    instr;  
    instr;  
}
```

```
if (wyr)  
    instr;  
else  
{  
    instr;  
    instr;  
}
```

Język C - Operatory relacyjne (porównania)

Operator	Przykład	Znaczenie
>	a > b	a większe od b
<	a < b	a mniejsze od b
>=	a >= b	a większe lub równe b
<=	a <= b	a mniejsze lub równe b
==	a == b	a równe b
!=	a != b	a nierówne b (a różne od b)

■ Wynik porównania jest wartością typu `int` i jest równy:

- **1** - gdy warunek jest prawdziwy
- **0** - gdy warunek jest fałszywy (nie jest prawdziwy)

Język C - Operatory logiczne

Operator	Znaczenie	Opis
!	NOT, nie	jednoargumentowy operator negacji logicznej - zmienia argument różny od zera na wartość 0, a argument równy zero na wartość 1
&&	AND, i	dwuargumentowy operator koniunkcji, iloczyn logiczny
	OR, lub	dwuargumentowy operator alternatywy, suma logiczna

■ Wynikiem zastosowania operatorów logicznych `&&` i `||` jest wartość typu `int` równa 1 (prawda) lub 0 (fałsz)

```
if (x>5 && x<8)
```

```
if (x<=5 || x>8)
```

Język C - Wyrażenia logiczne

- Wyrażenia logiczne mogą zawierać:

- operatory relacyjne
- operatory logiczne
- operatory arytmetyczne
- operatory przypisania
- zmienne
- stałe
- wywołania funkcji
- ...

- Kolejność operacji wynika z **priorytetu operatorów**

Operator	Typ operatora
!	logiczny
* / %	arytmetyczne
+ -	arytmetyczne
> < >= <=	relacyjne
== !=	relacyjne
&&	logiczny
	logiczny
=	przypisania

Język C - Wyrażenia logiczne

```
int x = 0, y = 1, z = 2;
```

```
if ( x == 0 )
```

wynik: 1 (prawda)

```
if ( x = 0 )
```

wynik: 0 (fałsz) (!!!)

```
if ( x != 0 )
```

wynik: 0 (fałsz)

```
if ( x != 0 )
```

wynik: 1 (prawda) (!!!)

```
if ( z > x + y )
```

wynik: 1 (prawda)

```
if ( z > ( x + y ) )
```

Język C - Wyrażenia logiczne

```
int x = 0, y = 1, z = 2;
```

```
if ( x>2 && x<5 )
```

wynik: 0 (fałsz)

```
if ( ( x>2 ) && ( x<5 ) )
```

- Wyrażenia logiczne obliczane są od strony lewej do prawej
- Proces obliczeń kończy się, gdy wiadomo, jaki będzie wynik całego wyrażenia

```
if ( 2 < x < 5 )
```

wynik: 1 (prawda) (!!!)

Język C - Wyrażenia logiczne

- W przypadku sprawdzania czy wartość wyrażenia jest równa lub różna od zera można zastosować skrócony zapis
- Zamiast:

```
if ( x == 0 )
```

```
if ( x != 0 )
```

można napisać:

```
if ( !x )
```

```
if ( x )
```


Język C - Operator warunkowy

- Operator warunkowy składa się z dwóch symboli i trzech operandów

```
wyrażenie1 ? wyrażenie2 : wyrażenie3
```

- Najczęściej zastępuje proste instrukcje `if-else`

```
float akcyza, cena, pojemnosc;
```

```
if (pojemnosc <= 2000)
    akcyza = cena*0.031; /* 3.1% */
else
    akcyza = cena*0.186; /* 18.6% */
```

```
akcyza = pojemnosc <= 2000 ? cena*0.031 : cena*0.186;
```

Język C - Operator warunkowy

```
if (x < 0)
    y = -x;
else
    y = x;
```

```
y = x < 0 ? -x : x;
```

- obliczenie modułu liczby `x`

```
if (a > b)
    max = a;
else
    max = b;
```

```
max = a > b ? a : b;
```

- wyznaczenie `max` z dwóch liczb

- Operator warunkowy ma bardzo niski priorytet
- Niższy priorytet mają tylko operatory przypisania (`=`, `+=`, `-=`, ...) i operator przecinkowy (`,`)

Język C - Operator warunkowy

- `x` studentów chce dojechać z akademika do biblioteki - ile taksówek powinni zamówić? (jedna taksówka może przewieźć 4 osoby)

```
#include <stdio.h>

int main(void)
{
    int x, taxi;

    printf("Podaj liczbę studentów: ");
    scanf("%d", &x);

    taxi = x / 4 + (x % 4 ? 1 : 0);

    printf("Liczba taxi: %d\n", taxi);

    return 0;
}
```

```
Podaj liczbę studentów: 23
Liczba taxi: 6
```

Język C - Instrukcja switch

- Instrukcja wyboru wielowariantowego `switch`

```
switch (wyrażenie)
{
    case wyrażenie_stale: instrukcje;
    case wyrażenie_stale: instrukcje;
    case wyrażenie_stale: instrukcje;
    ...
    default: instrukcje;
}
```

- `wyrażenie_stale` - wartość typu całkowitego, znana podczas kompilacji
 - stała liczbowa, np. 3, 5, 9
 - znak w apostrofach, np. 'a', 'z', '+'
 - stała zdefiniowana przez `const` lub `#define`

Język C - Instrukcja switch

- Program wyświetlający słownie liczbę z zakresu 1..5 wprowadzoną z klawiatury

```
#include <stdio.h>

int main(void)
{
    int liczba;

    printf("Podaj liczbę (1..5): ");
    scanf("%d", &liczba);
}
```

Język C - Instrukcja switch

```
switch (liczba)
{
    case 1: printf("Liczba: jeden\n");
            break;
    case 2: printf("Liczba: dwa\n");
            break;
    case 3: printf("Liczba: trzy\n");
            break;
    case 4: printf("Liczba: cztery\n");
            break;
    case 5: printf("Liczba: piec\n");
            break;
    default: printf("Inna liczba\n");
}
}
```

Podaj liczbę: 2
Liczba: dwa

Podaj liczbę: 0
Inna liczba

Język C - Instrukcja switch

```
switch (liczba)
{
    case 1:
    case 3:
    case 5: printf("Liczba nieparzysta\n");
            break;
    case 2:
    case 4: printf("Liczba parzysta\n");
            break;
    default: printf("Inna liczba\n");
}
}
```

Podaj liczbę: 2
Liczba parzysta

- Te same instrukcje mogą być wykonane dla kilku etykiet `case`

Język C - Instrukcja switch

```
switch (liczba)
{
    case 1: case 3: case 5:
            printf("Liczba nieparzysta\n");
            break;
    case 2: case 4:
            printf("Liczba parzysta\n");
            break;
    default: printf("Inna liczba\n");
}
}
```

Podaj liczbę: 2
Liczba parzysta

- Etykiety `case` mogą być pisane w jednym wierszu

Język C - Instrukcja switch

```
switch (liczba%2)
{
    case 1: case -1:
        printf("Liczba nieparzysta\n");
        break;
    case 0:
        printf("Liczba parzysta\n");
}
```

Podaj liczbe: 2
Liczba parzysta

- Część domyślna (**default**) może być pominięta

Język C - Instrukcja switch (bez break)

```
switch (liczba)
{
    case 1: printf("Liczba: jeden\n");
    case 2: printf("Liczba: dwa\n");
    case 3: printf("Liczba: trzy\n");
    case 4: printf("Liczba: cztery\n");
    case 5: printf("Liczba: piec\n");
    default: printf("Inna liczba\n");
}
```

Podaj liczbe: 2
Liczba: dwa
Liczba: trzy
Liczba: cztery
Liczba: piec
Inna liczba

- Pominięcie instrukcji **break** spowoduje wykonanie wszystkich instrukcji występujących po danym **case** (do końca **switch**)

Język C - suma kolejnych 10 liczb: 1+2+...+10

```
#include <stdio.h>

int main(void)
{
    int suma;

    suma = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

    printf("Suma wynosi: %d\n", suma);

    return 0;
}
```

Suma wynosi: 55

Język C - suma kolejnych 100 liczb: 1+2+...+100

```
#include <stdio.h>

int main(void)
{
    int suma=0, i;

    for (i=1; i<=100; i=i+1)
        suma = suma + i;

    printf("Suma wynosi: %d\n", suma);

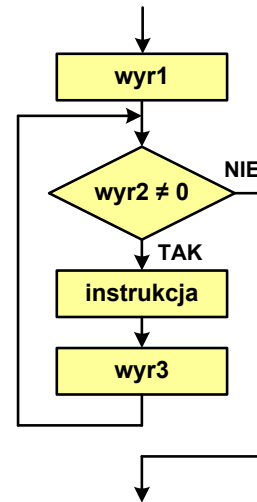
    return 0;
}
```

Suma wynosi: 5050

Język C - pętla for

```
for (wyr1; wyr2; wyr3)  
instrukcja
```

- **wyr1, wyr2, wyr3** - dowolne wyrażenia w języku C
- Instrukcja:
 - **prosta** - jedna instrukcja zakończona średnikiem
 - **złożona** - jedna lub kilka instrukcji objętych nawiasami klamrowymi



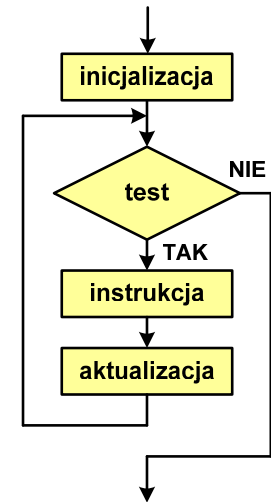
Język C - pętla for

- Najczęściej stosowana postać pętli **for**

```
int i;  
for (i = 0; i < 10; i = i + 1)  
instrukcja
```

- Instrukcja zostanie wykonana 10 razy (dla $i = 0, 1, 2, \dots, 9$)
- Funkcje pełnione przez wyrażenia

```
for (inicjalizacja; test; aktualizacja)  
instrukcja
```



Język C - pętla for (wyświetlenie tekstu)

```
#include <stdio.h>  
  
int main(void)  
{  
    int i;  
  
    for (i=0; i<5; i=i+1)  
        printf("Programowanie nie jest trudne\n");  
  
    return 0;  
}
```

```
Programowanie nie jest trudne  
Programowanie nie jest trudne  
Programowanie nie jest trudne  
Programowanie nie jest trudne  
Programowanie nie jest trudne
```

Język C - pętla for (suma liczb: 1 + 2 + ... + N)

```
#include <stdio.h>  
#define N 1234  
  
int main(void)  
{  
    int i, suma=0;  
  
    for (i=1; i<=N; i++)  
        suma = suma + i;  
  
    printf("Suma %d liczb to %d\n", N, suma);  
  
    return 0;  
}
```

```
Suma 1234 liczb to 761995
```

Język C - pętla for (przykłady)

```
for (i=0; i<10; i++)  
    printf("%d ",i);
```

0 1 2 3 4 5 6 7 8 9

```
for (i=0; i<10; i++)  
    printf("%d ",i+1);
```

1 2 3 4 5 6 7 8 9 10

```
for (i=1; i<=10; i++)  
    printf("%d ",i);
```

1 2 3 4 5 6 7 8 9 10

Język C - pętla for (przykłady)

```
for (i=1; i<10; i=i+2)  
    printf("%d ",i);
```

1 3 5 7 9

```
for (i=10; i>0; i--)  
    printf("%d ",i);
```

10 9 8 7 6 5 4 3 2 1

```
for (i=-9; i<=9; i=i+3)  
    printf("%d ",i);
```

-9 -6 -3 0 3 6 9

Język C - pętla for (break, continue)

- W pętli `for` można stosować instrukcje skoku: `break` i `continue`

```
int i;  
for (i=1; i<10; i++)  
{  
    if (i%2==0)  
        continue;  
    if (i%7==0)  
        break;  
    printf("%d\n",i);  
}
```

□ `continue` przerywa bieżącą iterację i przechodzi do obliczania `wyr3`

□ `break` przerywa wykonywanie pętli

1 3 5

Język C - pętla for (najczęstsze błędy)

- Postawienie średnika na końcu pętli `for`

```
int i;  
for (i=0; i<10; i++);  
    printf("%d ",i);
```

10

- Przecinki zamiast średników pomiędzy wyrażeniami

```
int i;  
for (i=0, i<10, i++)  
    printf("%d ",i);
```

Błąd kompilacji!

error C2143: syntax error : missing ';' before ')'

Język C - pętla for (najczęstsze błędy)

- Błędny warunek - brak wykonania instrukcji

```
int i;  
for (i=0; i>10; i++)  
    printf("%d ", i);
```

- Błędny warunek - pętla nieskończona

```
int i;  
for (i=1; i>0; i++)  
    printf("%d ", i);
```

```
1 2 3 4 5 6 7 8 9 ...
```

Język C - pętla nieskończona

```
for (wyr1; wyr2; wyr3)  
    instrukcja
```

- Wszystkie wyrażenia (**wyr1**, **wyr2**, **wyr3**) w pętli for są opcjonalne

```
for ( ; ; )  
    instrukcja
```

- pętla nieskończona

- W przypadku braku **wyr2** przyjmuje się, że jest ono **prawdziwe**

Język C - zagnieżdżanie pętli for

- Jako instrukcja w pętli **for** może występować kolejna pętla **for**

```
int i, j;  
for (i=1; i<=3; i++)           // pętla zewnętrzna  
    for (j=1; j<=2; j++)       // pętla wewnętrzna  
        printf("i: %d  j: %d\n", i, j);
```

```
i: 1  j: 1  
i: 1  j: 2  
i: 2  j: 1  
i: 2  j: 2  
i: 3  j: 1  
i: 3  j: 2
```

Język C - operator inkrementacji (++)

- Jednoargumentowy operator **++** zwiększa wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator **++** może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
++x	preinkrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
x++	postinkrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - operator inkrementacji (++)

- Przykład

```
int x = 1, y;  
y = 2 * ++x;
```

```
int x = 1, y;  
y = 2 * x++;
```

- Kolejność operacji

```
++x      x = 2  
2 * ++x  2 * 2  
y = 2 * ++x  y = 4
```

```
2 * x      2 * 1  
y = 2 * x  y = 2  
x++       x = 2
```

- Wartości zmiennych

```
x = 2    y = 4
```

```
x = 2    y = 2
```

Język C - operator dekrementacji (--)

- Jednoargumentowy operator -- zmniejsza wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator -- może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
--x	predekrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
x--	postdekrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - operator inkrementacji (++)

- Miejsce umieszczenia operatora ++ nie ma znaczenia w przypadku instrukcji typu:

```
x++;  
++x;
```

równoważne

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

- Nie należy stosować operatora ++ do zmiennych pojawiających się w wyrażeniu więcej niż jeden raz

```
x = x++;  
x = ++x;
```

- Zgodnie ze standardem języka C wynik powyższych instrukcji jest **niezdefiniowany**

Język C - priorytet operatorów ++ i --

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

Koniec wykładu nr 2

Dziękuję za uwagę!