Podstawy języka C – cz. I

dr inż. Maciej Kusy

Katedra Podstaw Elektroniki
Wydział Elektrotechniki i Informatyki
Politechnika Rzeszowska

Elektronika i Telekomunikacja, sem. 2

Plan wykładu

- Wprowadzenie do języka C
- Pojęcie zmiennej (właściwości)
- Typy danych
- Modyfikatory typów
- Operatory, priorytet operatorów
- Instrukcje sterujące:
  – warunkowe
  – iteracyjne (pętle)
- Formatowanie wejścia i wyjścia

Wprowadzenie do języka C

- C – strukturalny język programowania wysokiego poziomu stworzony przez Dennisa Ritchiego (pocz. lat 70-tych XX w.);
- Powstał poprzez rozwinięcie języka B (Thomson, Ritchie, 1969 r.);
- Oficjalna dokumentacja powstała w 1978 r. (Kernighan, Ritchie, pol. tł. *Język ANSI C*);
- W roku 1973 zaimplementowano jądro systemu operacyjnego Unix przy użyciu języka C;
- Na bazie języka C w latach osiemdziesiątych Bjarne Stroustrup stworzył język C++;
- W latach 80-tych i 90-tych XX w. C był dominującym językiem do tworzenia systemów operacyjnych i aplikacji.

Pojęcie zmiennej w języku C

Zmienna – instancja, element o ustalonej nazwie, typie i rozmiarze w obszarze (fragmencie) pamięci do przechowywania wartości, która zależy od typu zmiennej.

Zmienną należy zadeklarować – podać kompilatorowi jej nazwę i typ:

```
typ nazwa;
```

Zmiennej w momencie zadeklarowania można od razu przypisać wartość.

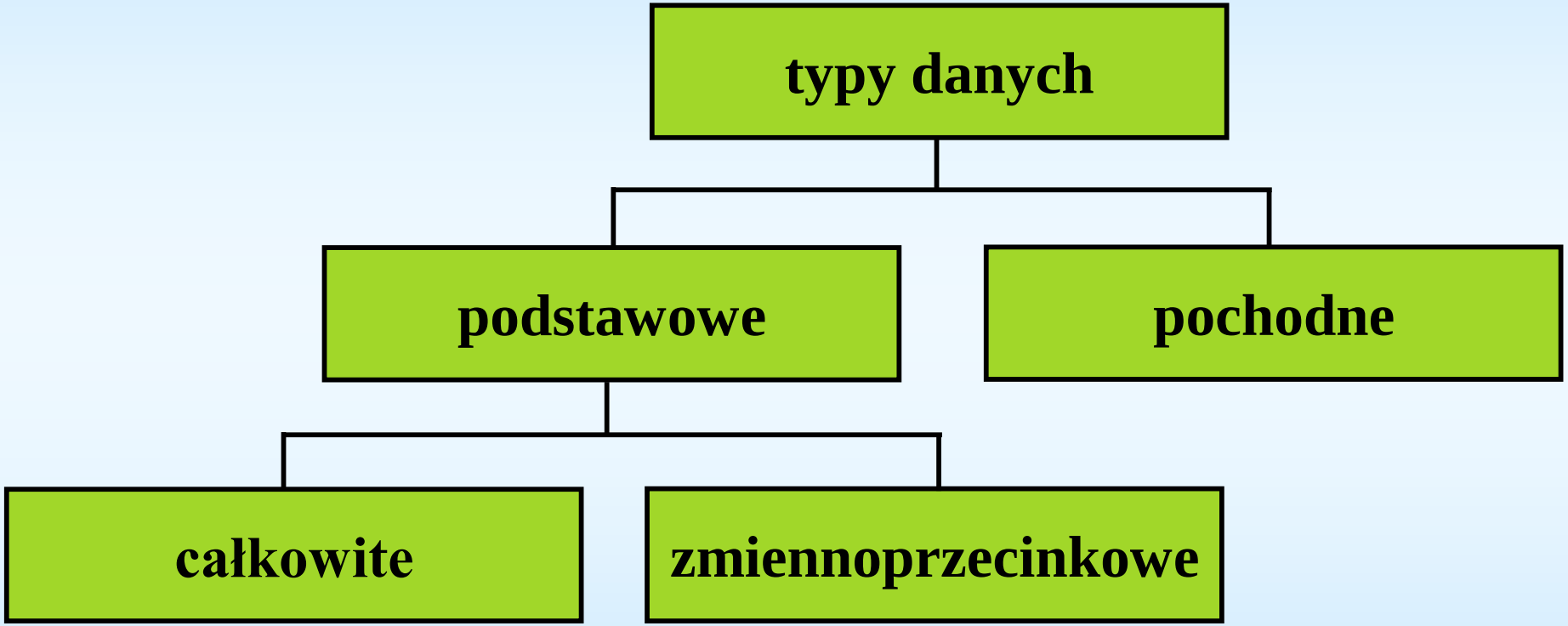
Zmienne mogą być dostępne dla wszystkich funkcji programu: **zmienne globalne**.

Zmienne dostępne w określonym zakresie (np. wewnątrz definicji funkcji): **zmienne lokalne**.

Właściwości zmiennych

- Posiadają typ oraz nazwę
- Na każdą zmienną danego typu przydzielona jest odpowiednia liczba bajtów
- Przyjmują określony zakres liczbowy
- W nazwie zmiennej nie mogą znajdować się znaki białe, symbole szczególne (operatory)
- Nazwa zmiennej nie może rozpoczynać się od cyfry
- Domyślna długość nazwy zmiennej: 256 znaków
- Można je rzutować z jednego typu na drugi
- Można je przekazywać do funkcji przez wartość lub poprzez wskaźnik
- Mogą mieć zasięg lokalny, globalny
- Mogą być składowymi struktur

Typy danych


typy danych
podstawowe
pochodne
całkowite
zmiennoprzecinkowe

Typy całkowite

Nazwa	Liczba bajtów	Zakres liczbowy
`char`	1	-128 ... 127
`short int`	2	-32768 ... 32767 (-2^{15} ... $2^{15}-1$)
`int`	4	-2 147 483 648 ... 2 147 483 647 (-2^{31} ... $2^{31}-1$)
`long int`	4	j.w.

Typy zmiennoprzecinkowe

Nazwa	Liczba bajtów	Zakres liczbowy	Precyzja
`float`	4	$\pm 3.4\ e^{\pm 38}$	7 cyfr znaczących
`double`	8	$\pm 1.7\ e^{\pm 308}$	15 cyfr
`long double`	10	$-3.4...1.1\ e^{\pm 4932}$	19 cyfr

Pozostałe typy

- Typ wyliczeniowy **`enum`**
- Typ pusty – **`void`**

Modyfikatory typów

- **`signed`** (domyślny typ całkowity)
- **`unsigned`**
- **`auto`** (domyślny zakres lokalny)
- **`register`**
- **`static`**
- **`volatile`**
- **`const`**
- **`extern`**

Pozostałe typy całkowite

Nazwa	Zakres liczbowy
`unsigned char`	0 ... 255
`unsigned short int`	0 ... 65 535 (0 ... 2^{16}-1)
`unsigned int`	0 .. 4 294 967 295 (0 ... 2^{32}-1)
`unsigned long int`	j.w.

Operatory arytmetyczne

```
+   -   *   /   %   ++   --
```

Przykład:

```
double delta = b*b-4.0*a*c;
```

Przykłady dzielenia:

`7/2` wynosi **3** (dzielenie całkowite)

`7.0/2` wynosi **3.5** (dzielenie zmiennoprzecinkowe)

`7%2` wynosi **1** (reszta z dzielenia)

Operatory arytmetyczne – cd.

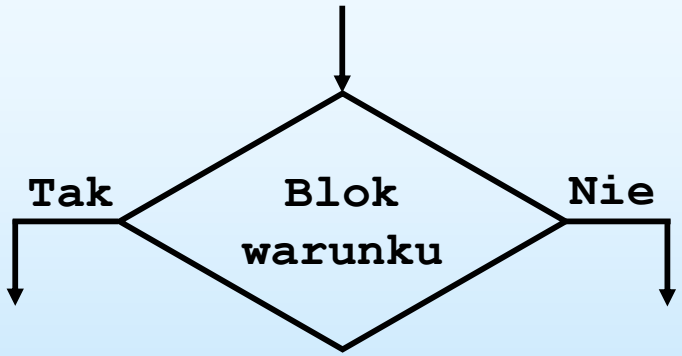
Operatory inkrementacji - przykłady

```
int i=0, j;
i++;      =>    i=i+1;

i=1;
j=i++;   => j=i; i=i+1;   czyli j=1, i=2

i=1;
j=++i;   => i=i+1; j=i;   czyli i=2, j=2
```

Operatory porównania (relacji)

```
>   <   >=   <=   ==   !=
```

Zastosowanie – instrukcje warunkowe:



Operatory logiczne

Iloczyn: **`&&`** Suma: **`||`** Negacja: **`!`**

Przykłady:

```
int x = 1, y = 7;
```

`x == 3 && y == 7` **FAŁSZ**

`x == 3 || y == 7` **PRAWDA**

`!(x == 3)` **PRAWDA**

Warunek na rok przestępny dla zmiennej **`rok`**:

```
(rok%4==0)&&(rok%100!=0)||(rok%400==0)
```

Operatory bitowe

& | ~ ^ >> <<

```
int a = 1234, b = 3456, c;
```

Rozpisując na bity:

```
a =>  0000 0100 1101 0010
b =>  0000 1101 1000 0000

c = a & b;     //0000 0100 1000 0000
c = a | b;     //0000 1101 1101 0010
c = a ^ b;     //0000 1001 0101 0010
c = a << 1;    //0000 1001 1010 0100
c = a >> 2;    //0000 0001 0011 0100
```

Operatory przypisania

=

+= -= *= /= %= &= |= ^= <<= >>=

Przykłady:

```
a = b = c;      //(a = (b = c));


a += b;   //a = a + b;
a >>= b;  //a = a >> b;
```

Priorytet operatorów

Priorytet	Symbol	Nazwa	Łączność
15	`.`	wybór składowej	L
	`->`	wybór składowej	
	`[]`	indeksowanie	
	`()`	wywołanie funkcji	
	`()`	nawias w wyrażeniach	
14	`sizeof`	rozmiar w bajtach obiektu lub typu	P
	`++`	inkrementacja (post i pre)	
	`--`	dekrementacja (post i pre)	
	`~`	negacja (not) bitowa	
	`!`	negacja logiczna	
	`-`	jednoargumentowy minus	
	`+`	jednoargumentowy plus	
	`&`	pobranie adresu (jednoargumentowy)	
	`*`	odniesienie się do elementu wskazywanego przez wskaźnik	
	`()`	konwersja (rzutowanie) wartości	

Priorytet operatorów – cd.

Priorytet	Symbol	Nazwa	Łączność

13	`*`
	`/`
	`%`
		mnożenie
		dzielenie
		reszta z dzielenia (modulo)
			L

12	`+` `-`	dodawanie odejmowanie	L
11	`<<` `>>`	przesunięcie bitowe w lewo przesunięcie bitowe w prawo	L

10	`<`
	`<=`
	`>`
	`>=`
		mniejsze niż
		mniejsze lub równe
		większe od
		większe lub równe
			L

9	`==` `!=`	równe różne	L
8	`&`	iloczyn (and) bitowy	L
7	`^`	różnica symetryczna (exor) bitowy	L
6	`|`	suma (or) bitowa	L

Priorytet operatorów – cd.

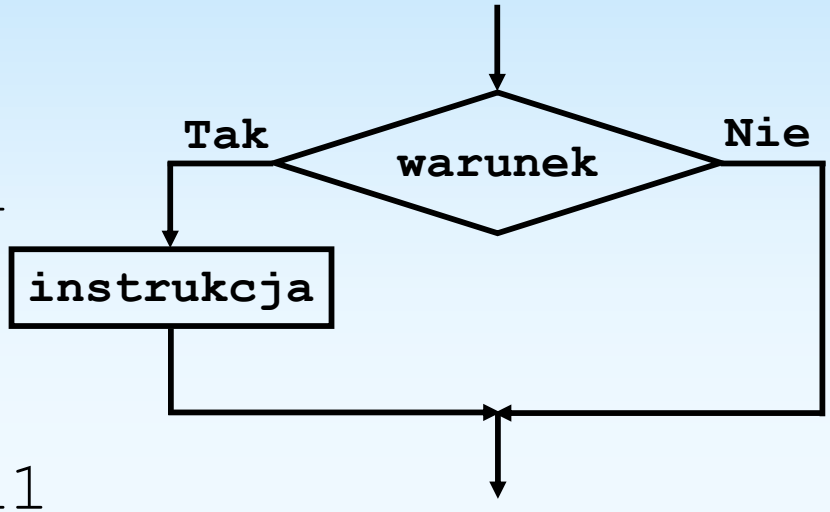
Priorytet	Symbol	Nazwa	Łączność
5	`&&`	koniunkcja (iloczyn logiczny)	L
4	`||`	alternatywa (suma logiczna)	L
3	`?:`	arytmetyczne wyrażenie warunkowe	L

2	`=`
	`*=`
	`/=`
	`%=`
	`+=`
	`-=`
	`<<=`
	`>>=`
	`&=`
	`|=`
	`^=`
		przypisanie
		pomnóż i przypisz
		podziel i przypisz
		policz resztę z dzielenia i przypisz
		dodaj i przypisz
		odejmij i przypisz
		przesuń bitowo w lewo i przypisz
		przesuń bitowo w prawo i przypisz
		policz iloczyn bitowy i przypisz
		policz sumę bitową i przypisz
		policz bitową różnicę symetryczną i przypisz
			P

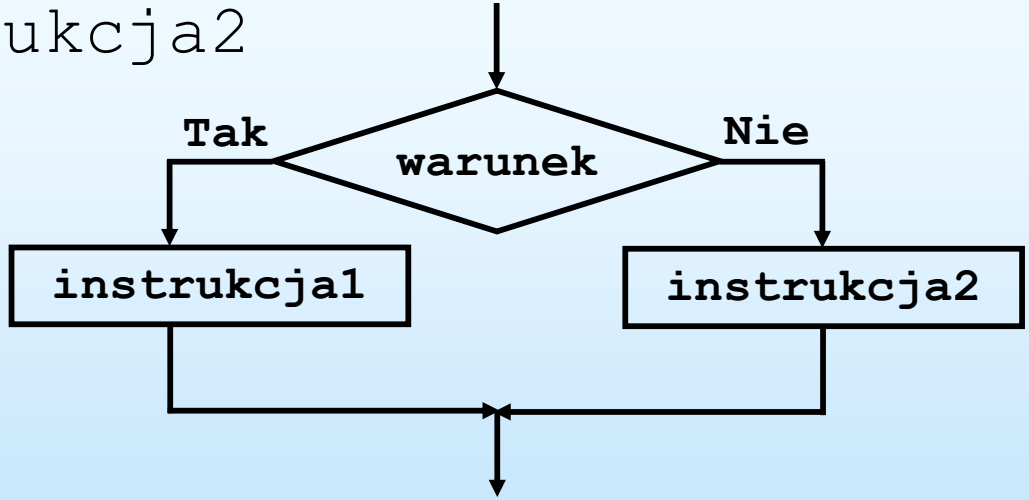
1	`,`	przecinek	L

Instrukcje warunkowe `if`, `if else`, `if else if`

```
if (warunek) instrukcja
```




```
if (warunek) instrukcja1
else         instrukcja2
```



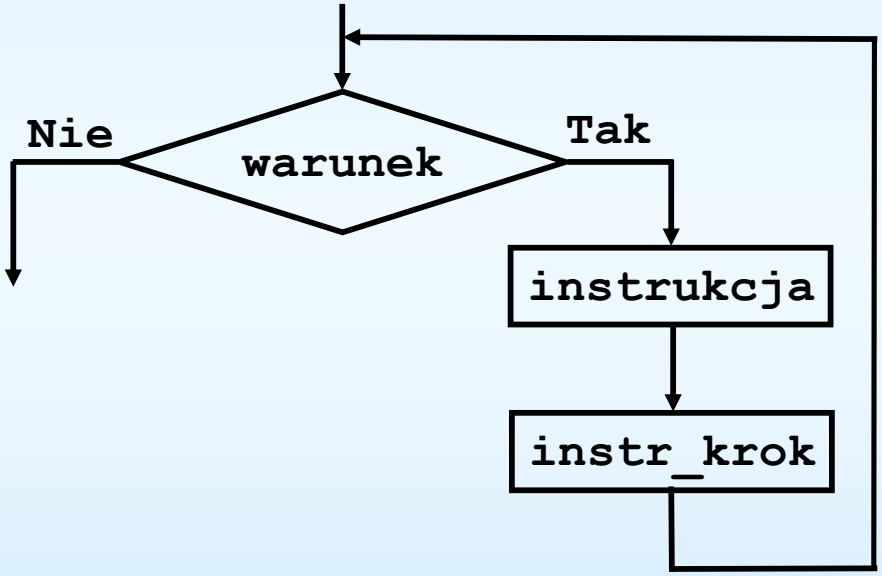

```
if else if
```

Instrukcja warunkowe **switch**

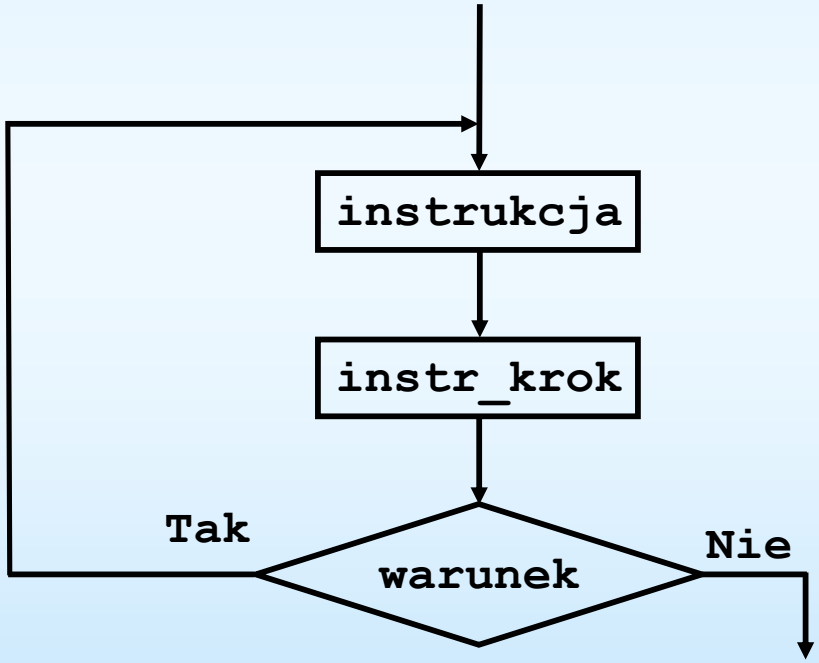
```
switch(wyrażenie)
{
  case wartość_wyrażenia_1:
     instrukcja1;
     break;
  case wartość_wyrażenia_2:
     instrukcja2;
     break;
  default:
     instrukcja_dla_pozostałych_wartości;
     break;
}
```

Pętle **while** i **do while**

```
while (warunek)
      instrukcja
```

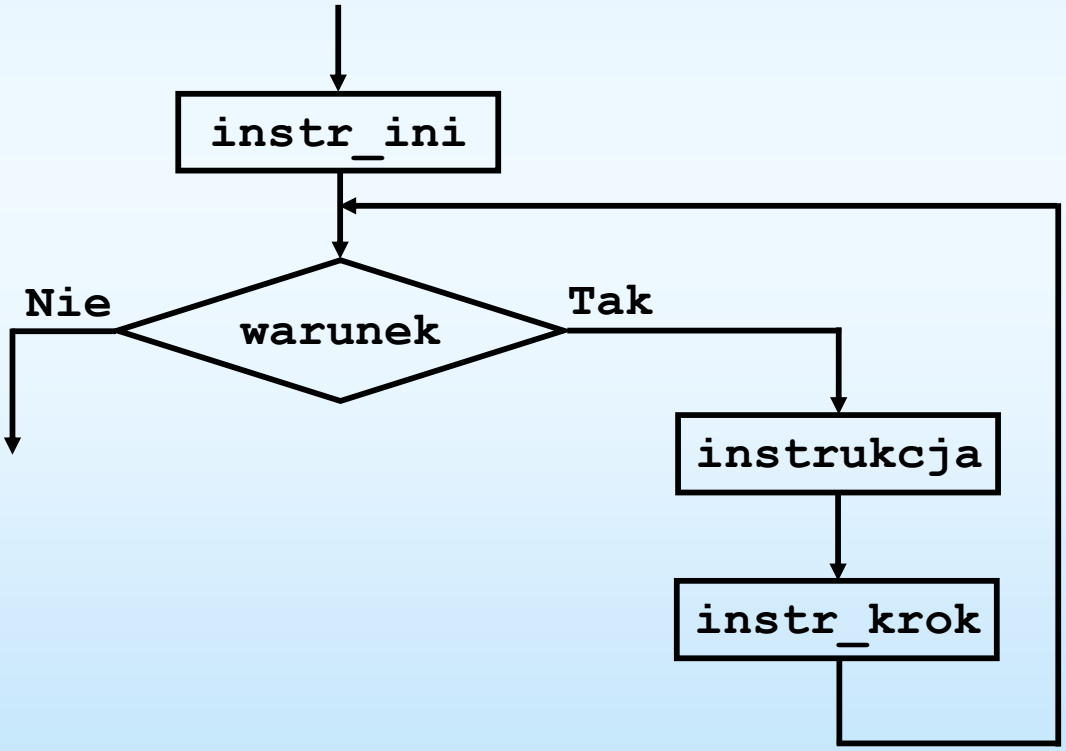



```
do
   instrukcja
while (warunek);
```



Pętla **for**

```
for (inst_ini; warunek; inst_krok)
     instrukcja
```



Pozostałe instrukcje sterujące

- **goto**

```
goto etykieta;
etykieta:
```

- **break**
- **continue**

Formatowanie wyjścia - `printf`

- Specyfikacja przekształcenia - **%**
- Znaki przekształcenia:
  - **`d`** lub **`i`** - argument wyświetlany w formie dziesiętnej
  - **`o`** - argument wyświetlany w formie ósemkowej (bez **0**)
  - **`x`** - argument wyświetlany w formie szesnastkowej (bez **0x**)
  - **`u`** - argument wyświetlany w formie dziesiętnej bez znaku
  - **`c`** - argument wyświetlany jako znak
  - **`s`** - argument wyświetlany jako łańcuch znakowy
  - **`e`** - argument wyświetlany jako liczba **`float`** lub **`double`** w postaci naukowej [-]m.nnnnnnE[±]xx
  - **`f`** - argument wyświetlany jako liczba **`float`** lub **`double`** w postaci stało przecinkowej [-]mmm.nnnnnn

Formatowanie wejścia - `scanf`

- Specyfikacja przekształcenia - **%**
- Znaki przekształcenia:
  - **`d`** lub **`i`** - argument pobierany jest liczbą całkowitą
  - **`o`** - argument pobierany jest liczbą całkowitą w formie ósemkowej (bez **0**)
  - **`x`** - argument pobierany jest liczbą całkowitą w formie szesnastkowej (bez **0x**)
  - **`h`** - argument pobierany jest liczbą całkowitą krótką (**`short int`**)
  - **`c`** - argument pobierany jest znakiem
  - **`s`** - argument pobierany jest łańcuchem znakowym
  - **`f`** - argument pobierany jest liczbą typu **`float`**
  - **`lf`** - argument pobierany jest liczbą typu **`double`**

Formatowanie wejścia – wyjścia. Przykład: