Politechnika Wrocławska

Wstęp do programowania

INP001213Wcl

rok akademicki 2018/19

semestr zimowy

Wykład 2

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

A-1 p. 411B

Politechnika Wrocławska

Plan prezentacji (1)

- Algorytmy liniowe
- Algorytmy z rozgałęzieniami

Na podstawie:

- M. M. Sysło, *Algorytmy*

Politechnika Wrocławska

Plan prezentacji (2)

- Podstawy języka C
  - komentarze
  - funkcja printf()
  - zmienne (typy danych)
  - łańcuchy znakowe
  - dyrektywy preprocesora
  - interakcja z użytkownikiem
  - wybrane operatory
  - instrukcja **`if`**
  - instrukcja **`if...else`**

Na podstawie:

- G. Perry, D. Miller, *Język C Programowanie dla początkujących*

Politechnika Wrocławska

Algorytmy liniowe

- Algorytm liniowy ma postać listy kroków, które mają być wszystkie wykonane zgodnie z kolejnością w jakiej występują.

Politechnika Wrocławska

Algorytmy liniowe

- Obliczanie wartości wielomianu

$$w(x) = ax^2 + bx + c$$

$$w(x) = (ax + b)x + c$$

$$v(x) = ax^3 + bx^2 + cx + d$$

$$v(x) = ((ax + b)x + c)x + d$$

Politechnika Wrocławska

Algorytmy z rozgałęzieniami

- Zwykle wykonywanie algorytmu uzależnione jest od spełnienia pewnych warunków.

Politechnika Wrocławska

Komentarze

- Komentarz – informacja umieszczona w kodzie źródłowym programu, która objaśnia jego działanie
- Komentarze są ignorowane przez kompilator
- Dodawaj komentarze podczas pisania programu

Politechnika Wrocławska

Komentarze

- Kod bez komentarza

```
return ((s1 < s2) ? s1 : s2);
```

- Kod z komentarzem

```
return ((s1 < s2) ? s1 : s2); /* znajduje
        mniejszą z dwóch wartości*/
```

- Kod z nadmiarowym komentarzem

```
printf("Lista płac"); /*drukuje napis
        "Lista płac"*/
```

Komentarze

Przykład kodu z komentarzami

Start here | wdp_l02p01.c

```c
/*wdp_l02p01.c*/
/*Na podstawie:
  G. Perry, D. Miller,
  Język C Programowanie dla początkujących,
  Helion, 2014*/

/*Oblicza łączny koszt prezentów*/
#include <stdio.h>
int main(){
   /*zmienne do przechowywania cen*/
   float gift1, gift2, gift3, gift4;
   float total; /*zmienna do przechowywania sumy*/

   /*pytania o ceny prezentów*/
   printf("Ile kosztuje prezent dla mamy? ");
   scanf(" %f",&gift1);
   printf("Ile kosztuje prezent dla taty? ");
   scanf(" %f",&gift2);
   printf("Ile kosztuje prezent dla siostry? ");
   scanf(" %f",&gift3);
   printf("Ile kosztuje prezent dla brata? ");
   scanf(" %f",&gift4);

   total = gift1+gift2+gift3+gift4; /*sumuje wszystkie ceny*/
   printf("\nNa prezenty wydasz %.2f PLN", total);
   return 0;
}
```

Politechnika Wrocławska

Komentarze

- W trakcie pracy nad programem można wykorzystywać komentarze do wyłączania pewnych fragmentów kodu

Politechnika Wrocławska

Komentarze

- Komentarz jednowierszowy rozpoczyna się znacznikiem `//`
- Obejmuje wszystko co za tym znacznikiem do końca wiersza

Start here | wdp_l02p02.c

```
1  //wdp_l02p02.c
2  //kod z innym rodzajem komentarza
3  #include <stdio.h>
4  main()
5  {
6     printf("Nowe komentarze!"); //prosta instrukcja
7     return 0;
8  }
9
```

Funkcja `printf()`

- Funkcja **`printf()`** wysyła dane wyjściowe na ekran
- Wypisywane mogą być znaki, liczby, słowa

Start here | wdp_l01p02.c

```
1   #include <stdio.h>
2
3   main(){
4       printf("To jest program w %c.\n",'C');
5       printf("Laboratorium %d z WdP.\n",1);
6       printf("WdP na %.1f ",99.9);
7       printf("procent.\n");
8       return 0;
9   }
10
```

Politechnika Wrocławska

Funkcja `printf()`

- Ogólny schemat wywołania funkcji **`printf()`**:
  `printf(`*`łańcuchKontrolny`*` [, `*`dane`*`]);`
- Przykład użycia
  `printf("Moja ulubiona liczba to %d", 7);`
- Na końcu każdego polecenia musi znajdować się średnik
- Funkcja **`printf()`** wysyła dane na standardowe wyjście

Politechnika Wrocławska

Funkcja `printf()`

Drukowanie łańcuchów

- Łańcuchy znakowe są najłatwiejsze do wypisania

```
printf("Jakie to proste!");
```

Funkcja `printf()`

Znaki specjalne

kod	opis
`\n`	nowa linia
`\a`	alarm
`\b`	backspace
`\t`	tabulator
`\\`	lewy ukośnik
`\'`	pojedynczy cudzysłów
`\"`	podwójny cudzysłów

Funkcja `printf()`

Znaki specjalne - przykład

Start here | printf_znaki_specjalne.c

```c
#include <stdio.h>

main(){
    printf("Kolumna A\tKolumna B\tKolumna C\n");
    printf("To jest apostrof: \'\n");
    printf("Alarm!!\a\n");
    printf("Cudzyslow podwojny \"\n");
    printf("Samolot\b\b\bwar\n");
    printf("Backslash: \\\n");
    return 0;
}
```

```
Kolumna A       Kolumna B       Kolumna C
To jest apostrof: '
Alarm!!
Cudzyslow podwojny "
Samowar
Backslash: \
```

Funkcja `printf()`

Znaczniki konwersji

- W celu wypisania liczb i znaków, należy użyć znaczników konwersji

znacznik konwersji	opis
`%d`	liczba całkowita
`%f`	liczba zmiennopozycyjna
`%c`	znak
`%s`	łańcuch

Politechnika Wrocławska

Funkcja `printf()`

Znaczniki konwersji

- W miejscu, w którym ma być wypisana wartość, należy użyć odpowiedniego znacznika konwersji
- Jako kolejne argumenty funkcji **`printf()`** należy podać wartości do wstawienia

Funkcja `printf()`

Znaczniki konwersji - przykład

```
printf("Cena rogala to %d groszy, zatem %d
kosztuja %.2f zlotego",60,3,1.8);
```

```
Cena rogala to 60 groszy, zatem 3 kosztuja
1.80 zlotego
```

Politechnika Wrocławska

Funkcja `printf()`

Znaczniki konwersji - przykład

```
printf("%s %d %f %c\n", "WdP", 13, 3.2, 'Q');
```

```
WdP 13 3.200000 Q
```

Funkcja `printf()`

Znaczniki konwersji

- Format drukowanych danych można kontrolować wpisując kropkę i liczbę w znacznik konwersji

```
printf("%f  %.3f  %.2f  %.1f",
4.5678, 4.5678, 4.5678, 4.5678);

4.567800  4.568  4.57  4.6
```

Politechnika Wrocławska

Zmienne

- Zmienna to miejsce w pamięci komputera, w którym zapisano znak lub liczbę
- Ponieważ rozróżniamy różne typy danych, zatem rozróżniamy różne typy zmiennych
- W danej zmiennej nie można zapisać danych dowolnego rodzaju

Politechnika Wrocławska

Zmienne

Najczęściej używane typy danych

nazwa	opis
`char`	typ reprezentujący dane znakowe, np. `'A'` i `'#'`
`int`	typ reprezentujący dane całkowite
`float`	typ reprezentujący dane zmiennopozycyjne
`double`	typ reprezentujący dane zmiennopozycyjne o podwójnej precyzji

Politechnika Wrocławska

Zmienne

Nadawanie nazw

- Nazwy zmiennych nie mogą się powtarzać
- Nazwa zmiennej może mieć od 1 do 31 znaków
- Nazwa zmiennej powinna zaczynać się literą, po której mogą znajdować się inne litery, liczby lub znak podkreślenia

Zmienne

Deklarowanie zmiennych

- Deklaracja zmiennej to konstrukcja języka C, informująca, że potrzebujemy zarezerwować obszar pamięci na zmienną określnego typu

```
main(){
    //deklaracje zmiennych
    char odpowiedz;
    int liczba;
    float cena;
    /* pozostała część programu */
```

Politechnika Wrocławska

Zmienne

Deklarowanie zmiennych

- W jednym wierszu można zadeklarować kilka zmiennych tego samego typu

```
main(){
    //deklaracje zmiennych
    int x;
    int y;
    /* pozostała część programu */

main(){
    //deklaracje zmiennych
    int x, y;
    /* pozostała część programu */
```

Politechnika Wrocławska

Zmienne

Deklarowanie zmiennych

- Większość deklaracji zmiennych znajduje się za klamrą otwierającą funkcję – są to zmienne lokalne
- Można też tworzyć zmienne globalne, ale prawie zawsze należy się posługiwać zmiennymi lokalnymi

Politechnika Wrocławska

Zmienne

Operator przypisania

- W celu wstawienia danych do zmiennych (zdefiniowania) używa się operatora przypisania

```
zmienna = dane;
```

Zmienne

Operator przypisania

- Zmienna powinna już być zadeklarowana, aby można było do niej coś przypisać

```
main(){
    //deklaracje zmiennych
    char odpowiedz;
    int liczba;
    float cena;
    //przypisanie wartości
    odpowiedz = 'C';
    liczba = 16;
    cena = 9.99;
    /* pozostała część programu */
```

Politechnika Wrocławska

Zmienne

Operator przypisania

- W zmiennych można też zapisywać wyniki wyrażeń

```
 //oblicza cenę z 25% rabatem
cena = 9.99 * .75;

 //oblicza łączną cenę produktów
 razem = cena * liczba;
```

Zmienne

Przykład

Start here | celsjusz_fahrenheit.c

```
1   /*celsjusz_fahrenheit.c*/
2   /*przeliczanie temperatury
3     ze stopni Celsjusza
4     na stopnie Fahrenheita*/
5   main(){
6       float celsjusz, fahrenheit;
7       celsjusz = 36.6;
8       fahrenheit = 32 + 1.8*celsjusz;
9       printf("Temperatura %.2f stopni Celsjusza\n",celsjusz);
10      printf("odpowiada temperaturze %.2f stopni Fahrenheita.", fahrenheit);
11      return 0;
12  }
13
```

Politechnika Wrocławska

Łańcuchy znakowe

- Na końcu każdego łańcucha dodawany jest znak specjalny – znak końca łańcucha
- Znak końca łańcucha `'\0'` nie jest znakiem `'0'`
- Program wykorzystuje znak `'\0'` do rozpoznawania końca łańcucha

`"WdP"`

W	d	P	\0

Łańcuchy znakowe

Długość łańcucha

- Długość łańcucha to liczba znaków do znaku końca łańcucha (wliczając spacje i inne białe znaki)

- Łańcuch:
  `Wstep do programowania`
  ma 22 znaki

Łańcuchy znakowe

Znak a łańcuch o jednym znaku

'x'

"x"

Politechnika Wrocławska

Łańcuchy znakowe

Tablice znaków

- Do przechowywania łańcuchów w pamięci służą tablice znaków
- Tablica danych danego typu to ciąg zmiennych w pamięci
- Deklaracja tablicy:
  ***typ nazwa*[*rozmiar*];**

Łańcuchy znakowe

Tablice znaków

- Deklaracja tablicy:
  ***typ nazwa*[*rozmiar*];**
- Przykład:

```
char miesiac[12];
/*najdłuższa nazwa miesiąca
   to październik
   ma 11 znaków, ale potrzebujemy
   miejsce na znak końca łańcucha*/
```

Łańcuchy znakowe

Tablice znaków

- Można zainicjalizować tablicę w miejscu jej deklaracji

```
char miesiac[12] = "styczen";
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
s	t	y	c	z	e	n	\0	?	?	?	?

Łańcuchy znakowe

Tablice znaków - `printf()`

- Do wypisywania tablicy znaków służy znacznik konwersji `%s`

```
printf("Teraz jest %s", miesiac);
```

Łańcuchy znakowe

Tablice znaków - inicjalizacja

- W przypadku jednoczesnej deklaracji i inicjalizacji nie trzeba podawać rozmiaru tablicy, ale nie będzie można wpisać w tą tablicę dłuższego łańcucha

```
char miesiac[8] = "styczen";
char miesiac[] = "styczen";

//więcej miejsca na dłuższe łańcuchy
char miesiac[25] = "styczen";
```

Łańcuchy znakowe

Tablice znaków - przypisanie

- Do tablicy nie można przypisać łańcucha operatorem przypisania

```
miesiac = "lipiec"; /*źle*/
```

- Można przypisać litera po literze

```
miesiac[0] = 'l';
miesiac[1] = 'i';
miesiac[2] = 'p';
miesiac[3] = 'i';
miesiac[4] = 'e';
miesiac[5] = 'c';
miesiac[6] = '\0';
```

Łańcuchy znakowe

Tablice znaków - `strcpy()`

- Do wpisania łańcucha znakowego do tablicy można użyć funkcji **`strcpy()`**

```
strcpy(miesiac,"lipiec");
```

- Aby móc skorzystać z tej funkcji należy dołączyć plik nagłówkowy **`<string.h>`**

```
#include <string.h>
```

Politechnika Wrocławska

Dyrektywy preprocesora

- Dyrektywy preprocesora nie wpływają bezpośrednio na działanie programu
- Dyrektywy preprocesora są wykorzystywane w procesie kompilacji
- Najczęściej wykorzystywanymi dyrektywami są:
  - **`#include`**
  - **`#define`**

Politechnika Wrocławska

Dyrektywa `#include`

- Dyrektywa **`#include`** występuje w dwóch formach:

  `#include <`*`nazwapliku`*`>`

  `#include "`*`nazwapliku`*`"`

- Dyrektywa **`#include`** łączy pliki przed kompilacją

Politechnika Wrocławska

Dyrektywa #include

Start here | main.c | included.h

```c
#include <stdio.h>

main(){
    printf("1. To jest napis z funkcji main().\n");
    #include "included.h"
    printf("2. To jest napis z funkcji main().\n");
    return 0;
}
```

main.c

Start here | main.c | included.h

```c
printf("To jest napis z dolaczonego pliku.\n");
```

included.h

Start here | main.c

```c
#include <stdio.h>

main(){
    printf("1. To jest napis z funkcji main().\n");
    printf("To jest napis z dolaczonego pliku.\n");
    printf("2. To jest napis z funkcji main().\n");
    return 0;
}
```

to widzi kompilator

Politechnika Wrocławska

Dyrektywa #include

Start here | main.c | included.h

```c
#include <stdio.h>

main(){
    printf("1. To jest napis z funkcji main().\n");
    #include "included.h"
    printf("2. To jest napis z funkcji main().\n");
    return 0;
}

```

main.c

Start here | main.c | included.h

```c
printf("To jest napis z dolaczonego pliku.\n");

```

included.h

```
1. To jest napis z funkcji main().
To jest napis z dolaczonego pliku.
2. To jest napis z funkcji main().
```

Politechnika Wrocławska

Dyrektywa `#include`

- Do standardowych zadań wykorzystuje się wbudowane pliki biblioteczne
- Dyrektywa `#include <nazwapliku>` służy do dołączania wbudowanych plików

Politechnika Wrocławska

Dyrektywa `#include`

- Dotychczas wmienione były funkcje:
  - **`printf()`** plik nagłówkowy **`stdio.h`**
  - **`strcpy()`** plik nagłówkowy **`string.h`**

Politechnika Wrocławska

Dyrektywa `#include`

- Programista może opracować własne pliki biblioteczne
- Dyrektywa **`#include "`*`nazwapliku`*`"`** służy do dołączania plików nagłówkowych
- W tym przypadku w pierwszej kolejności przeszukiwany jest katalog, w którym znajduje się program, a potem katalog bibliotek wbudowanych

Politechnika Wrocławska

Definiowanie stałych

- Dyrektywa **`#define`** służy do definiowania stałych
  `#define` ***`STAŁA definicjaStałej`***

- Przykłady:

```
#define LIMITWIEKU 18
#define WYDZIAL "PPT"
#define PI 3.14159
```

Definiowanie stałych

- Każde wystąpienie nazwy stałej zastępowane jest jej definicją

```
//definicja stałej
#define LIMITWIEKU 18

//wiersz programu
if(wiekPracownika < LIMITWIEKU)

//kompilator interpretuje jako
if(wiekPracownika < 18)
```

Politechnika Wrocławska

Dyrektywy preprocesora

- Dyrektywy preprocesora nie są poleceniami języka C
- Dyrektywy są przetwarzane przed kompilacją

Politechnika Wrocławska

Interakcja z użytkownikiem

- Funkcja **`printf()`** służy do wyświetlenia danych na ekranie
- Do pobrania danych z klawiatury służy funkcja **`scanf()`**

Politechnika Wrocławska

Interakcja z użytkownikiem

- Funkcje **`printf()`** i **`scanf()`** zdefiniowane są w tym samym pliku nagłówkowym **`stdio.h`**
- Funkcja **`scanf()`** wykorzystuje znaczniki konwersji (np. **`%s`**, **`%d`**)

Politechnika Wrocławska

Interakcja z użytkownikiem

- Ogólny format funkcji **`scanf()`**
  `scanf(`*`łańcuchKontrolny`*` [, `*`dane`*`]);`

- Pobieranie wartości przez funkcję **`scanf()`** kończy się po naciśnięciu klawisza Enter

Politechnika Wrocławska

Interakcja z użytkownikiem

- Prawie zawsze wywołanie funkcji **`scanf()`** poprzedzone powinno być wywołaniem funkcji **`printf()`**
- Wywołanie funkcji **`printf()`** służy do wyświetlenia pytania lub instrukcji dla użytkownika jakie dane powinien podać

Politechnika Wrocławska

Interakcja z użytkownikiem

- Przykładowe wywołanie funkcji **`scanf()`**

```
printf("Jaka jest pierwsza litera Twojego
imienia?\n");
scanf(" %c",&firstInitial);
```

Interakcja z użytkownikiem

Przykład (1)

Start here | scanf_example.c

```c
/*scanf_example.c*/
/*na podstawie: G. Perry, D. Miller,
Język C Programowanie dla początkujących, Helion, 2014*/

main(){
    char firstInitial;
    char lastInitial;
    int age;
    int favouriteNumber;

    printf("Jaka jest pierwsza litera Twojego imienia?\n");
    scanf(" %c",&firstInitial);

    printf("Jaka jest pierwsza litera Twojego nazwiska?\n");
    scanf(" %c",&lastInitial);

    printf("Ile masz lat?\n");
    scanf(" %d",&age);
```

Interakcja z użytkownikiem

Przykład (2)

Start here | scanf_example.c

```
11      printf("Jaka jest pierwsza litera Twojego imienia?\n");
12      scanf(" %c",&firstInitial);
13
14      printf("Jaka jest pierwsza litera Twojego nazwiska?\n");
15      scanf(" %c",&lastInitial);
16
17      printf("Ile masz lat?\n");
18      scanf(" %d",&age);
19
20      printf("Jaka jest Twoja ulubiona liczba (tylko liczby calkowite)?\n");
21      scanf(" %d",&favouriteNumber);
22
23      printf("\nTwoje inicjaly to %c.%c. i masz %d lat.",
24             firstInitial, lastInitial, age);
25      printf("\nTwoja ulubiona liczba to %d.\n\n",
26             favouriteNumber);
27
28      return 0;
29  }
```

Politechnika Wrocławska

Interakcja z użytkownikiem

- Funkcja **`scanf()`** wymaga, aby przed nazwą każdej zmiennej znajdował się znak **`&`**
- Wyjątkiem jest wczytywanie łańcucha znaków znacznikiem **`%s`**

Obliczenia matematyczne

Podstawowe działania arytmetyczne

```
sprzedazOgolem = sprzedazKraj + sprzedazUE - zwroty;

printf("Za trzy lata będę mieć %d lat.\n", age + 3);

cenaBrutto = cenaNetto * 1.23;

skladka = kosztCalkowity / liczbaOsob;
//Uwaga na dzielenie całkowitoliczbowe!
```

Obliczenia matematyczne

Operatory dzielenia

Start here | operatory.c

```
1  /*operatory.c*/
2  /*Karol Tarnowski*/
3  /*Wstęp do programowania*/
4  /*Przykładowy program demonstrujący
5    działanie operatorów dzielenia
6    i dzielenie modulo.*/
7  /*na podstawie: G. Perry, D. Miller,
8  Język C Programowanie dla początkujących, Helion, 2014*/
9
10 #include <stdio.h>
11
12 main(){
13     /*Dwa zestawy równoważnych zmiennych:*/
14     /*zmiennopozycjnych*/
15     float xf = 22.0;
16     float yf =  5.0;
17     float wynikFloat;
18     /*całkowioliczbowych*/
19     int xi = 22;
20     int yi = 5;
21     int wynikInt;
```

Obliczenia matematyczne

Operatory dzielenia

Start here | operatory.c

```
23      //Iloraz dwóch liczb zmiennopozycyjnych wyniesie 4.4
24      wynikFloat = xf/yf;
25      printf("%.1f podzielone na %.1f wynosi %.1f.\n", xf, yf, wynikFloat);
26
27      //Iloraz dwóch liczb całkowitoliczbowych wyniesie 4
28      wynikInt = xi/yi;
29      printf("%.1d podzielone na %.1d wynosi %.1d.\n", xi, yi, wynikInt);
30
31      //W tym przypadku zachodzi obcięcie wartości
32      wynikInt = xf/yf;
33      printf("%.1f podzielone na %.1f wynosi %.1d.\n", xf, yf, wynikInt);
34
35      //Wykorzystując operator modulo (%) można obliczyć resztę
36      wynikInt = xi%yi;
37      printf("Resza dzielenia %.1d przez %.1d wynosi %.1d.\n", xi, yi, wynikInt);
38
39      return 0;
40  }
```

Obliczenia matematyczne

Kolejność wykonywania działań (1/2)

operator	łączność
`()` (nawias) `[]` (element tablicy), `.` (odwołanie do składowej struktury)	lewostronna
`-` (znak ujemności), `++` (inkrementacja), `--` (dekrementacja), `&` (adres), `*` (wskaźnik), `sizeof()`, `!` (negacja)	prawostronna
`*` (mnożenie), `/` (dzielenie), `%` (modulo)	lewostronna
`+` (dodawanie), `-` (odejmowanie)	lewostronna
`<` (mniejszość), `<=` (mniejszy/równy), `>` (większość), `>=` (większy równy)	lewostronna

Obliczenia matematyczne

Kolejność wykonywania działań (2/2)

operator	łączność
`==` (równość), `!=` (nierówność)	lewostronna
`&&` (logiczne i)	lewostronna
`||` (logiczne lub)	lewostronna
`? :` (operator warunkowy)	prawostronna
`=, *=, /=, %=, +=, -=` (operatory przypisania)	prawostronna
`,` (przecinek)	lewostronna

Politechnika Wrocławska

Obliczenia matematyczne

Kolejność wykonywania działań

```
/*zły sposób obliczenia
  średniej*/
srednia = i + j + k + l / 4;

/*tak jest dobrze*/
srednia = (i + j + k + l) / 4;
```

Politechnika Wrocławska

Obliczenia matematyczne

Kolejność wykonywania działań

```
ans = 5 + 2 * 3;
/*wynikiem wykonania instrukcji
  jest przypisanie do zmiennej
  ans wartości 11*/

ans = (5 + 2) * 3;
/*jednak tym razem wynikiem
  wykonania instrukcji jest
  przypisanie do zmiennej ans
  wartości 21*/
```

Politechnika Wrocławska

Operator przypisania

```
/*możliwy przykład przypisania dziesięciu
  zmiennym wartości 9*/
a = 9; b = 9; c = 9; d = 9; e = 9;
f = 9; g = 9; h = 9; i = 9; j = 9;

/*zestaw równoważnych instrukcji zapisanych
  z wykorzystaniem prawostronnej łączności
  operatora przypisania*/
a = b = c = d = e = f = g = h = i = j = 9;
```

Politechnika Wrocławska

Operator przypisania

```
/*W języku C zmienne nie są inicjalizowane
  automatycznie. Jeśli trzeba zainicjalizować
  kilka zmiennych wartością 0, to można
  wykorzystać przypisanie zbiorcze*/

a = b = c = d = e = f = g = h = i = j = 0;
```

Politechnika Wrocławska

Operator przypisania

```
/*W języku C każde wyrażenie zwraca jakąś
  wartość. Można to wykorzystać,
  w niespodziewany sposób. */
a = 2 * (b = 5);

/*Równoważny fragment kodu.*/
b = 5;
a = 2 * b;
```

Politechnika Wrocławska

Rzutowanie typów

- Rzutowanie typów polega na tymczasowej zmianie typu danych zmiennej.

`(typDanych) wartość`

- Typem danych może być każdy typ języka C (np. **`int`** lub **`float`**). Wartością może być zmienna, literał, wyrażenie.

Politechnika Wrocławska

Rzutowanie typów

```
int wiek = 19;

//Wyrażenie poniżej ma wartość 19.0
(float)wiek

rabat = cena * (float)wiek / 100.0;
```

- Rzutowanie nie zmienia typu zmiennej. Działa tylko w miejscu użycia.

Politechnika Wrocławska

Testowanie danych

- Instrukcje **`if`** lub **`if...else`** wykorzystuje się do uzależnienia biegu działania programu od danych
- Do oceny prawdziwości warunków wykorzystać można operatory relacyjne

Politechnika Wrocławska

Testowanie danych

operator	relacja
==	równe
<	mniejsze
>	większe
<=	mniejsze lub równe
>=	większe lub równe
!=	różne

Politechnika Wrocławska

Testowanie danych

```
int i = 5;
int j = 10;
int k = 15;
int l = 5;

//instrukcje prawdziwe
i == l;
j < k;
k > i;
j != l;
```

Politechnika Wrocławska

Testowanie danych

```
int i = 5;
int j = 10;
int k = 15;
int l = 5;

//instrukcje fałszywe
i > j;
k < j;
k == l;
```

Politechnika Wrocławska

Testowanie danych

- Operatory relacyjne zwracają 1 gdy warunek jest spełniony i 0 gdy warunek jest fałszywy

```
a = ( 4 < 10 ); // a = 1;
b = ( 8 == 9 ); // b = 0;
```

Politechnika Wrocławska

Instrukcja `if`

- Ogólna postać

```
if(warunek)
{
   blok instrukcji C;
}
```

Politechnika Wrocławska

Instrukcja `if`

- Ogólna postać

```
if(warunek) //warunek objęty nawiasem
{
   blok instrukcji C;
}
/*klamra jest konieczna, gdy blok
instrukcji zawiera więcej niż jedną
instrukcję*/
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe.c

```
1  /*rownanie_liniowe.c*/
2  /*Karol Tarnowski*/
3  /*Wstęp do programowania*/
4  /*Program rozwiązujący równanie liniowe
5  (bez sprawdzenia poprawności danych)*/
6  #include <stdio.h>
7
8  main(){
9      float a, b;
10
11     printf("Program rozwiazuje rownanie a*x + b = 0\n");
12     printf("dla podanych wspolczynnikow a oraz b.\n\n");
13
14     //wczytywanie danych
15     printf("Podaj a: ");
16     scanf("%f",&a);
17     printf("Podaj b: ");
18     scanf("%f",&b);
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe.c

```
19
20      //obliczenie wyniku i wyświetlenie rozwiązania
21      printf("Rozwiazaniem jest x = %g\n",-b/a);
22      return 0;
23  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 2
Podaj b: -4
Rozwiazaniem jest x = 2
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe.c

```
19
20      //obliczenie wyniku i wyświetlenie rozwiązania
21      printf("Rozwiazaniem jest x = %g\n",-b/a);
22      return 0;
23  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 0
Podaj b: 4
Rozwiazaniem jest x = -1.#INF
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe.c

```
19
20      //obliczenie wyniku i wyświetlenie rozwiązania
21      printf("Rozwiazaniem jest x = %g\n",-b/a);
22      return 0;
23  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 0
Podaj b: 0
Rozwiazaniem jest x = -1.#IND
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe_if.c

```
1  /*rownanie_liniowe_if.c*/
2  /*Karol Tarnowski*/
3  /*Wstęp do programowania*/
4  /*Program rozwiązujący równanie liniowe
5  (sprawdza, czy a != 0)*/
6
7  #include <stdio.h>
8
9  main(){
10    float a, b;
11
12    printf("Program rozwiazuje rownanie a*x + b = 0\n");
13    printf("dla podanych wspolczynnikow a oraz b.\n\n");
14
15    //wczytywanie danych
16    printf("Podaj a: ");
17    scanf("%f",&a);
18    printf("Podaj b: ");
19    scanf("%f",&b);
20
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe_if.c

```
21      //sprawdzenie czy a != 0
22      if(a == 0){
23          printf("Program nie dziala poprawnie dla a = 0.\n");
24          printf("Podaj a: ");
25          scanf("%f",&a);
26      }
27
28      //obliczenie wyniku i wyświetlenie rozwiązania
29      printf("Rozwiazaniem jest x = %g\n",-b/a);
30      return 0;
31  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 0
Podaj b: -4
Program nie dziala poprawnie dla a = 0.
Podaj a: 2
Rozwiazaniem jest x = 2
```

Politechnika Wrocławska

Instrukcja `if`

Start here | rownanie_liniowe_if.c

```
21      //sprawdzenie czy a != 0
22      if(a == 0){
23          printf("Program nie dziala poprawnie dla a = 0.\n");
24          printf("Podaj a: ");
25          scanf("%f",&a);
26      }
27
28      //obliczenie wyniku i wyświetlenie rozwiązania
29      printf("Rozwiazaniem jest x = %g\n",-b/a);
30      return 0;
31  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 0
Podaj b: -4
Program nie dziala poprawnie dla a = 0.
Podaj a: 0
Rozwiazaniem jest x = 1.#INF
```

Instrukcja `if`

Start here | rownanie_liniowe_if.c

```
21      //sprawdzenie czy a != 0
22      if(a == 0){
23          printf("Program nie dziala poprawnie dla a = 0.\n");
24          printf("Podaj a: ");
25          scanf("%f",&a);
26      }
27
28      //obliczenie wyniku i wyświetlenie rozwiązania
29      printf("Rozwiazaniem jest x = %g\n",-b/a);
30      return 0;
31  }
```

Istnieje możliwość pytania użytkownika dopóki nie poda prawidłowych danych - pętla **`do...while`**

Politechnika Wrocławska

Instrukcja `if...else`

- Ogólna postać

```
if(warunek)
{
   blok instrukcji C;
}
else
{
   blok instrukcji C;
}
```

Politechnika Wrocławska

Instrukcja `if...else`

Start here | rownanie_liniowe_if_else.c

```
1  /*rownanie_liniowe_if_else.c*/
2  /*Karol Tarnowski*/
3  /*Wstęp do programowania*/
4  /*Program rozwiązujący równanie liniowe
5  (wykorzystuje instrukcję if...else)*/
6
7  #include <stdio.h>
8
9  main(){
10     float a, b;
11
12     printf("Program rozwiazuje rownanie a*x + b = 0\n");
13     printf("dla podanych wspolczynnikow a oraz b.\n\n");
14
15     //wczytywanie danych
16     printf("Podaj a: ");
17     scanf("%f",&a);
18     printf("Podaj b: ");
19     scanf("%f",&b);
20
```

Politechnika Wrocławska

Instrukcja `if...else`

Start here | rownanie_liniowe_if_else.c

```
20
21      //jeśli a == 0
22      if(a == 0){
23          printf("Niestety nie potrafie rozwiazac rownania dla a = 0.\n");
24      }
25      //jeśli a != 0
26      else{
27          //obliczenie wyniku i wyświetlenie rozwiązania
28          printf("Rozwiazaniem jest x = %g\n",-b/a);
29      }
30      return 0;
31  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 0
Podaj b: 4
Niestety nie potrafie rozwiazac rownania dla a = 0.
```

Zagnieżdżone instrukcje `if, if...else`

Start here | rownanie_liniowe_zagniezdzenia.c

```
1  /*rownanie_liniowe_zagniezdzenia.c*/
2  /*Karol Tarnowski*/
3  /*Wstęp do programowania*/
4  /*Program rozwiązujący równanie liniowe*/
5
6  #include <stdio.h>
7
8  main(){
9      float a, b;
10
11     printf("Program rozwiazuje rownanie a*x + b = 0\n");
12     printf("dla podanych wspolczynnikow a oraz b.\n\n");
13
14     //wczytywanie danych
15     printf("Podaj a: ");
16     scanf("%f",&a);
17     printf("Podaj b: ");
18     scanf("%f",&b);
19
```

Zagnieżdżone instrukcje `if, if...else`

Start here | rownanie_liniowe_zagniezdzenia.c

```
20      //jeśli a == 0
21      if(a == 0){
22          //jeśli a == 0 i b == 0
23          //to równanie jest prawdziwe tożsamościowo
24          if(b == 0){
25              printf("Rownanie jest prawdziwe\
26      dla wszystkich liczb rzeczywistych.\n");
27          }
28          //jeśli a == 0, a b != 0
29          //to równanie jest sprzeczne
30          else{
31              printf("Rownanie nie ma rozwiazan.\n");
32          }
33      }
34      //jeśli a != 0
35      else{
36          //obliczenie wyniku i wyświetlenie rozwiązania
37          printf("Rozwiazaniem jest x = %g\n",-b/a);
38      }
39      return 0;
40  }
```

Zagnieżdżone instrukcje `if, if...else`

Start here | rownanie_liniowe_zagniezdzenia.c

```
20      Program rozwiazuje rownanie a*x + b = 0
21      dla podanych wspolczynnikow a oraz b.
22
23      Podaj a: 0
24      Podaj b: 0
25      Rownanie jest prawdziwe dla wszystkich liczb rzeczywistych.
26  dl
27          }
28          //jeśli a == 0
29          //to równanie
30          else{
31              printf("Row
32          }
33      }
34      //jeśli a != 0
35      else{
36          //obliczenie w
37          printf("Rozwia
38      }
39      return 0;
40  }
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 0
Podaj b: 4
Rownanie nie ma rozwiazan.
```

```
Program rozwiazuje rownanie a*x + b = 0
dla podanych wspolczynnikow a oraz b.

Podaj a: 2
Podaj b: -4
Rozwiazaniem jest x = 2
```

Politechnika Wrocławska

Operatory logiczne

operator	relacja
&&	koniunkcja (i)
||	alternatywa (lub)
!	negacja

Politechnika Wrocławska

Operatory logiczne

Start here | rownanie_liniowe_operatory.c

```
1  /*rownanie_liniowe_operatory.c*/
2  /*Karol Tarnowski*/
3  /*Wstęp do programowania*/
4  /*Program rozwiązujący równanie liniowe
5  (wykorzystuje operatory logiczne)*/
6
7  #include <stdio.h>
8
9  main(){
10     float a, b;
11
12     printf("Program rozwiazuje rownanie a*x + b = 0\n");
13     printf("dla podanych wspolczynnikow a oraz b.\n\n");
14
15     //wczytywanie danych
16     printf("Podaj a: ");
17     scanf("%f",&a);
18     printf("Podaj b: ");
19     scanf("%f",&b);
20
```

Politechnika Wrocławska

Operatory logiczne

Start here | rownanie_liniowe_operatory.c

```
20
21      if(a == 0 && b == 0)//równanie tożsamościowe
22          printf("Rownanie jest prawdziwe dla wszystkich liczb rzeczywistych.\n");
23      else if(a == 0 && b != 0)//równanie sprzeczne
24          printf("Rownanie nie ma rozwiazan.\n");
25      else//w przeciwnym przypadku rozwiązanie jednoznaczne
26          printf("Rozwiazaniem jest x = %g\n",-b/a);
27
28      return 0;
29  }
```

Operatory logiczne

Kolejność wykonywania

- Operator `&&` ma wyższy priorytet niż operator `||`

```
if(  a < 20  ||    b < 120  &&  c > 15  )
```

- Powyższa instrukcja z nawiasami odpowiadającymi kolejności wykonywania działań

```
if( (a < 20) || ( (b < 120) && (c > 15) ) )
```

Politechnika Wrocławska

Podsumowanie

- Algorytmy liniowe
- Algorytmy z rozgałęzieniami
- Podstawy języka C
  - komentarze
  - funkcja printf()
  - zmienne (typy danych)
  - łańcuchy znakowe
  - dyrektywy preprocesora
  - interakcja z użytkownikiem
  - wybrane operatory
  - instrukcja `if`
  - instrukcja `if...else`