

Sprawozdanie z zajęć: „Podstawy Programowania” z dnia 27.10.2022r.

Laboratorium 5

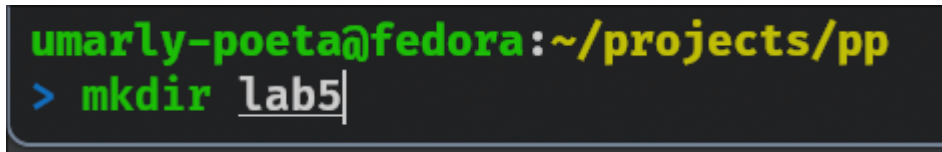
Imię i nazwisko: Patryk Kozłowski / Informatyka Techniczna

Temat: Konstrukcje sterujące if i switch w C

Cel: Opanowanie podstaw stosowania konstrukcji sterujących if i switch w C.

Opis:

1. Utworzyłem folder lab5



```
umarly-poeta@fedora:~/projects/pp  
> mkdir lab5|
```

2. utworzyłem folder switch_dir i skopiowałem simple_switch.c, następnie go skompilowałem i zamieniłem jego konstrukcje na postać standardową



```
[umarly-poeta:~/projects/pp/lab5]$ mkdir switch_dir| master
```

```

1  #include <stdio.h>
2
3  #define TOLERANCE 1
4
5  int main(void)
6  {
7
8      char c;
9      for(;;){ // wykonywanie nieskończenie wiele razy
10
11          printf("\nWprowadź cyfrę od 0 do 5 lub 6 aby zakonczyc program: \n");
12          scanf(" %c",&c); // wielokrotne wczytywanie pojedynczego znaku (jak pozbyć się \n ?)
13          switch (c) {
14              case '0':
15                  printf("Wprowadzono: 0\n");
16                  break;
17              case '1':
18                  printf("Wprowadzono: 1\n");
19              case '2':
20                  printf("Wprowadzono: 1 lub 2\n");
21                  break;
22              case '3':
23                  printf("Wprowadzono: 3\n");
24                  break;
25              case '4':
26              case '5':
27                  printf("Wprowadzono: 4 lub 5\n");
28                  break;
29              case '6':
30                  printf("pomyslnie zakonczono program");
31                  return 0;
32              default:
33                  printf("Wprowadzono: znak spoza zakresu 0-5\n");
34                  break;
35          }
36      }
37
38  }

```

```

[umaryl-poeta:~/projects/pp/lab5/switch_dir]$ ./switch
Wprowadź cyfrę od 0 do 5:
1
Wprowadzono: 1
Wprowadzono: 1 lub 2

Wprowadź cyfrę od 0 do 5:
2
Wprowadzono: 1 lub 2

Wprowadź cyfrę od 0 do 5:
3
Wprowadzono: 3

Wprowadź cyfrę od 0 do 5:
4
Wprowadzono: 4 lub 5

Wprowadź cyfrę od 0 do 5:
5
Wprowadzono: 4 lub 5

Wprowadź cyfrę od 0 do 5:
^C
[umaryl-poeta:~/projects/pp/lab5/switch_dir]$ █ (master*)

```

3. Napisałem identyczny program co `simple_switch.c` zmieniając konstrukcję `switch` na `if else`, następnie nazwałem go `simple_switch_as_if.c` i skompilowałem

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char c;
6      for(;;) {
7          printf("\nWprowadź cyfrę od 0 do 5 lub 6 aby zakonczyc program: \n");
8          scanf(" %c", &c);
9
10         if (c == '0') {
11             printf("Wprowadzono: 0\n");
12         } else if (c == '1') {
13             printf("Wprowadzono: 1\n");
14             printf("Wprowadzono: 1 lub 2\n");
15         } else if (c == '2') {
16             printf("Wprowadzono: 1 lub 2\n");
17         } else if (c == '3') {
18             printf("Wprowadzono: 3\n");
19         } else if (c == '4' || c == '5') {
20             printf("Wprowadzono: 4 lub 5\n");
21         } else if (c == '6') {
22             printf("pomyslnie zakonczono program");
23             return 0;
24         } else {
25             printf("Wprowadzono: znak spoza zakresu 0-5\n");
26         }
27     }
28 }
29
```

```
[umarly-poeta:~/projects/pp/leh5/switch_dir]$ ./to_z_ifami

Wprowadź cyfrę od 0 do 5:
1
Wprowadzono: 1
Wprowadzono: 1 lub 2

Wprowadź cyfrę od 0 do 5:
2
Wprowadzono: 1 lub 2

Wprowadź cyfrę od 0 do 5:
3
Wprowadzono: 3

Wprowadź cyfrę od 0 do 5:
4
Wprowadzono: 4 lub 5

Wprowadź cyfrę od 0 do 5:
5
Wprowadzono: 4 lub 5

Wprowadź cyfrę od 0 do 5:
^C
[umarly-poeta:~/projects/pp/leh5/switch_dir]$ [master*]
```

4. Uzupełniłem program tak, aby przed zakończeniem sprawdzał poprawność wyniku czy obliczone pierwiastki są rzeczywiście pierwiastkami ($ax^2 + bx + c = 0$), oraz zmodyfikowałem program rozwiązywania równania kwadratowego tak, aby rozwiązywał (z odpowiednimi komentarzami) przypadki równania liniowego ($\text{fabs}(a) < \text{TOLERANCJA}$) i o pierwiastkach zespolonych ($\Delta < 0$) oraz Modyfikacja programu `rownanie_kwadratowe.c` dla różnych przypadków kontraktu uwzględniającego skończoną precyzję obliczeń oraz Modyfikacja programu `rownanie_kwadratowe.c` w nowym pliku, np. `rownanie_kwadratowe_nieczytelne.c`) tak, aby wszystkie przypadki były uwzględnione w

jednej konstrukcji `if ... elsif ... elsif ... else` (wcięcie ma tylko jeden poziom, poprawność wymaga powtarzania pewnych operacji w różnych gałęziach) – Taki kod nie jest czytelny

```
[umarly-poeta:~/projects/pp/lab5]$ mkdir rownanie_kwadratowe|
```

```

1 #include <stdio.h> // ISES
2 #include <stdlib.h> // ISES
3 #include <math.h> // ISES
4
5 // sposób na łatwą zmianę typu dla wszystkich zmiennych w programie
6 #define SCALAR double
7 #define SCALAR float
8 #define TOLERANCJA 1e-20 // czy wystarczy jedna wartość dla float i dla double?
9 #define SMALL_NUMBER 1e-20 // czy wystarczy jedna wartość dla float i dla double?
10
11 // Testowanie: konstrukcji sterujących (a także nazw zmiennych).
12 // zwrócić wskazówki i czas życia
13 int main(void)
14 {
15     // rozwiązanie równania kwadratowego  $ax^2 + bx + c = 0$ 
16
17     printf("Program rozwiązywania równania kwadratowego  $ax^2 + bx + c = 0$ !\n");
18
19     // typ a, b, c ustalony poprzez symbol SCALAR
20     SCALAR a, b, c;
21     // rozwiązanie z konstrukcją... (try napisać skróconą wersję)
22     // input - uśrednienie na błędy wczytania danych
23     printf("Wprowadź parametr a: "); scanf("%lf", &a); // adres! (miejsce na wpisanie wartości)
24     printf("Wprowadź parametr b: "); scanf("%lf", &b); // uwaga: inny format dla float!
25     printf("Wprowadź parametr c: "); scanf("%lf", &c);
26
27     // if(a==0 && b==0) { // alternatywa: if(a==0 || b==0) - zależnie od kontekstu
28     if (fabs(a)-SMALL_NUMBER && fabs(b)-SMALL_NUMBER) { // poprawa
29
30         printf("Błąd! dane: a i b równe 0 (dłgt bliskie 0). Przerwanie programu.\n");
31         exit(-1);
32     }
33
34     else{
35
36         if(fabs(a) < TOLERANCJA) { // równanie liniowe
37             printf("Wprowadź x: %lf!\n", (-c / b));
38             //no może konstrukcja - czy należy rozwiązywać liczby a bliskie 0 ?
39
40         }
41         else{
42
43             SCALAR delta; // różnica wskazówek nazwy - powiązanie z czasem życia
44             delta = b*b - 4*a*c; // problem jeśli delta bliskie 0 (w skróconej wersji)
45             // ... // wtedy obliczenie b i sqrt(delta) traci cyfry znaczące
46
47             if(delta < 0){
48                 SCALAR real_part = -b / (2 * a);
49                 SCALAR imaginary_part = sqrt(-delta) / (2 * a);
50                 printf("Dwa pierwiastki zespolone: x1 = %lf + %lfj, x2 = %lf - %lfj!\n",
51                     real_part, imaginary_part, real_part, imaginary_part);
52             } else if (delta == 0){
53
54             } else if (delta > 0){
55                 printf("Dwa pierwiastki rzeczywiste: x = %lf!\n", -b/(2*a));
56             } else {
57
58             }
59
60             SCALAR temp = sqrt(delta);
61
62             SCALAR x1 = (-b - temp) / (2 * a);
63             SCALAR x2 = (-b + temp) / (2 * a);
64
65             if (fabs(a * x1 * x1 + b * x1 + c) > TOLERANCJA) {
66                 printf("Pierwiastek x1 = %lf nie spełnia równania w granicach tolerancji.\n", x1);
67             }
68             else if (fabs(a * x2 * x2 + b * x2 + c) > TOLERANCJA) {
69                 printf("Pierwiastek x2 = %lf nie spełnia równania w granicach tolerancji.\n", x2);
70             }
71             else{
72                 printf("Dwa pierwiastki rzeczywiste: x1 = %lf, x2 = %lf!\n",
73                     (-b-temp)/(2*a), (-b+temp)/(2*a));
74             } // znaczenie ścieżki i nazwa kolumnowa dla zwiększenia czytelności kodu !
75         }
76     }
77
78     for (int k = 2; k <= 10; k++) {
79         b = pow(10, k);
80         a = 1.0;
81         c = 1.0;
82
83
84         SCALAR delta = b * b - 4 * a * c;
85         SCALAR temp = sqrt(delta);
86         SCALAR x1 = (-b - temp) / (2 * a);
87         SCALAR x2 = (-b + temp) / (2 * a);
88
89         printf("\nrb = 10^%d:\n", k);
90         printf("x1 = %lf, x2 = %lf!\n", x1, x2);
91         printf("Sprawdzenie: f(x1) = %lf!\n", a * x1 * x1 + b * x1 + c);
92         printf("Sprawdzenie: f(x2) = %lf!\n", a * x2 * x2 + b * x2 + c);
93     }
94
95     return(0);
96 }

```

```

1 #include <math.h> // sqrt
2 #include <stdio.h> // printf
3 #include <math.h> // sqrt
4
5 // wyznaczenie rodzaju zmiennej typu dla wszystkich zmiennych w programie
6 #define SCALAR double
7 //definiowanie SCALAR float
8 #define TOLERANCJA 1.e-10
9 #define SMALL_NUMBER 1.e-20 // czy wystarczy jedna wartosc dla float i dla double?
10
11 // Testowanie konstrukcji sterujacych (a takze name zmiennych,
12 // zmiennych wdroznic i czasu dziala)
13 int main(void)
14 {
15     // rozwiązanie równania kwadratowego ax^2 + bx + c = 0
16
17     printf("Program rozwiązywania równania kwadratowego ax^2 + bx + c = 0\n");
18
19     // typ a, b, c ustalany poprzez symbol SCALAR
20     SCALAR a, b, c;
21     // rozwiązanie z konstrukcją... (czy naprawdę skorzysta z precyzji?)
22     // input - wspomnienie na błędy wczytywania danych
23     printf("Podaj parametry a: "; scanf("%lf", &a); // adres (miejsce na wpisanie wartości)
24     printf("Podaj parametry b: "; scanf("%lf", &b); // uwagi: ten format dla float!
25     printf("Podaj parametry c: "; scanf("%lf", &c);
26
27     // if(a==0 && b==0) { // alternatywa: if(a==0 || b==0) - zależnie od kontekstu
28     if (fabs(a) < SMALL_NUMBER && fabs(b) < SMALL_NUMBER) { // poprawnie
29
30         printf("Błąd! dane: a i b równe 0 (zbyt bliskie 0). Przerwanie programu.\n");
31         exit(-1);
32     }
33
34     // else {
35
36     if (fabs(a) < TOLERANCJA) { // równanie liniowe
37         printf("x wynosi %12le\n", (-c / b));
38         // czy może konstrukcja? - czy należy rozwiązywać liczby a bliskie 0?
39     }
40
41     // else {
42
43     SCALAR delta; // obliczenie wartości delta - powtarzanie z czasem działo
44     delta = b*b - 4*a*c; // problem: jeśli delta bliskie 0 (w skrajnym przypadku)
45     // ... // wtedy obliczanie k i sqrt(delta) traci cyfry znaczące
46
47     if (delta < 0) {
48         SCALAR real_part = -b / (2 * a);
49         SCALAR imaginary_part = sqrt(-delta) / (2 * a);
50         printf("Dwa pierwiastki zespolone: x1 = %12le + %12le*i, x2 = %12le - %12le*i\n",
51             real_part, imaginary_part, real_part, imaginary_part);
52     }
53     // else if (delta == 0) {
54
55     printf("Jeden pierwiastek rzeczywisty: x = %12le\n", -b / (2*a));
56
57     // else {
58
59     SCALAR temp = sqrt(delta);
60
61     SCALAR x1 = (-b - temp) / (2 * a);
62     SCALAR x2 = (-b + temp) / (2 * a);
63
64     if (fabs(a) * x1 * x1 + b * x1 + c > TOLERANCJA) {
65         printf("Pierwiastek x1 = %12le nie spełnia równania w granicach tolerancji.\n", x1);
66     }
67     // else if (fabs(a) * x2 * x2 + b * x2 + c > TOLERANCJA) {
68     printf("Pierwiastek x2 = %12le nie spełnia równania w granicach tolerancji.\n", x2);
69     }
70     // else {
71     printf("Dwa pierwiastki rzeczywiste: x1 = %12le, x2 = %12le\n",
72         (-b-temp)/(2*a), (-b+temp)/(2*a));
73     } // znaczenie wcięt i hasłowo klamrowych dla zwiększenia czytelności kodu!
74 }
75 }
76
77
78 for (int k = 2; k <= 10; k++) {
79     b = pow(10, k);
80     a = 1.0;
81     c = 1.0;
82
83     SCALAR delta = b*b - 4*a*c;
84     SCALAR temp = sqrt(delta);
85     SCALAR x1 = (-b - temp) / (2 * a);
86     SCALAR x2 = (-b + temp) / (2 * a);
87
88     printf("x1 = %12le, x2 = %12le\n", k);
89     printf("x1 = %12le, x2 = %12le\n", x1, x2);
90     printf("Sprawdzenie: f(x1) = %12le, a * x1^2 + b * x1 + c = %12le\n",
91         f(x1), a * x1 * x1 + b * x1 + c);
92     printf("Sprawdzenie: f(x2) = %12le, a * x2^2 + b * x2 + c = %12le\n",
93         f(x2), a * x2 * x2 + b * x2 + c);
94 }
95
96 for (int k = 2; k <= 10; k++) {
97
98 }
99
100 return(0);
101
102 }

```

```

[main@ip-pasta: ~]# cd /root/.ssh/; ssh -o StrictHostKeyChecking=no root@192.168.1.100
Program wyznajacy pierwiastki rownania kwadratowego ax^2 + bx + c = 0

Podaj parametr a: 1
Podaj parametr b: 2
Podaj parametr c: 3
Dwa pierwiastki zespolone: x1 = -1.000000000000e+00 + 1.414213562373e+00i, x2 = -1.000000000000e+00 - 1.414213562373e+00i

b = 10^2:
x1 = -9.999999999999e+01, x2 = -1.000100010002e-02
Sprawdzanie: f(x1)= 0.000000000000e+00
Sprawdzanie: f(x2) = 1.221245377080e-13

b = 10^3:
x1 = -9.999999999999e+02, x2 = -1.000010000125e-03
Sprawdzanie: f(x1)= 1.304153318209e-10
Sprawdzanie: f(x2) = -2.802180040470e-11

b = 10^4:
x1 = -9.999999999999e+03, x2 = -1.0000010001113e-04
Sprawdzanie: f(x1)= 0.000000000000e+00
Sprawdzanie: f(x2) = -1.117083073282e-09

b = 10^5:
x1 = -9.999999999999e+04, x2 = -1.000001327330e-05
Sprawdzanie: f(x1)= 0.000000000000e+00
Sprawdzanie: f(x2) = -3.304377228045e-07

b = 10^6:
x1 = -9.999999999999e+05, x2 = -1.0000017014403e-06
Sprawdzanie: f(x1)= 0.000000000000e+00
Sprawdzanie: f(x2) = -7.034402309907e-08

b = 10^7:
x1 = -1.000000000000e+07, x2 = -9.902131294330e-08
Sprawdzanie: f(x1)= -1.502200000000e-02
Sprawdzanie: f(x2) = 3.404045101440e-03

b = 10^8:
x1 = -1.000000000000e+08, x2 = -7.42530230024e-09
Sprawdzanie: f(x1)= 1.000000000000e+00
Sprawdzanie: f(x2) = 2.340410462070e-01

b = 10^9:
x1 = -1.000000000000e+09, x2 = 0.000000000000e+00
Sprawdzanie: f(x1)= 1.000000000000e+00
Sprawdzanie: f(x2) = 1.000000000000e+00

b = 10^10:
x1 = -1.000000000000e+10, x2 = 0.000000000000e+00
Sprawdzanie: f(x1)= 1.000000000000e+00
Sprawdzanie: f(x2) = 1.000000000000e+00
[main@ip-pasta: ~]# cd /root/.ssh/; ssh -o StrictHostKeyChecking=no root@192.168.1.100

```

