Kompilacja i uruchomienie programu w szczegółach

Aby pracować podczas programowania efektywnie, warto zdawać sobie sprawę ze sposobu kompilacji i uruchomienia programów, rodzajów pojawiających się w tym procesie błędów i ostrzeżeń oraz możliwych do wykorzystania opcji. Na przykładzie prostego programu przedstawimy tu proces jego kompilacji i uruchomienia w środowisku dostępnym na serwerach panamint i diablo. W celu zapoznania się ze sposobami komunikowania błędów i ostrzeżeń przez kompilatory oraz zagadnieniem przenośności oprogramowania zaleca się przeprowadzenie kompilacji programu na obu systemach.

1. Środowisko pracy

Panamint jest komputerem HP ProLiant DL585 G7 firmy Hewlett-Packard, działającym pod nadzorem systemu linux z dystrybucji Debian GNU/Linux 12 (bookworm). Na komputerze tym dosttępny jest kompilator języka C w wersji GNU C 12.2.0 (uruchamiany poleceniem `gcc`). Diablo jest komputerem Sun Fire V880 firmy Sun Microsystems, działającym pod nadzorem systemu UNIX w wersji SunOS 5.10. Na komputerze tym dosttępne są dwa kompilatory języka C: w wersji GNU C 5.5.0 (uruchamiany poleceniem `gcc`) oraz w wersji Sun C 5.12 SunOS_sparc (uruchamiany poleceniem `cc`).

2. Działania przygotowawcze

Celem sprawnego przeprowadzenia całego procesu kompilacji i uruchomienia przykładowego programu warto otworzyć dwie konsole tekstowe i zalogować się z nich na serwerach panamint i diablo, co powinno przynieść efekt pokazany na wydrukach 1 i 2[1].

```
krecik@limba:~:)ssh panamint.kcir.pwr.edu.pl -X
Linux panamint 6.1.0-22-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.94-1 (2024-06-21) x86_64
------------------------------------------------------------------------------
Debian GNU/Linux 12 (bookworm)                          HP ProLiant DL585 G7
******************************************************************************
*                                                                            *
*                               P A N A M I N T                              *
*                        Debian GNU/Linux 12 (bookworm)                      *
*                                                                            *
*                             przegladarki www: w3m,lynx,links2,firefox      *
*                             klienty pocztowe: alpine,mutt                  *
*                               edytory tekstu: vi,vim,nano,emacs            *
*                           programy inzynierskie: NumPy/SciPy,octave        *
*                                                  mathematica,matlab        *
*                      narzedzia programistyczne: gcc,git,svn,designer       *
* narzedzie do tworzenia dokumentow tekstowych: LaTeX,TeXstudio              *
*                                pakiety biurowe: libreoffice                *
*                               menedzery plikow: mc,ranger                  *
*                                                                            *
*              W przypadku problemow prosimy pisac na adres:                 *
*                             admin@kcir.pwr.edu.pl                          *
******************************************************************************
You have new mail.
Last login: Fri Oct 18 08:35:38 2024 from 156.17.9.3
krecik@panamint:~:
```

Wydruk 1: Terminal użytkownika zalogowanego na panamincie

[1]W przypadku logowania się na panamincie/diablo na konto użytkownika o innej nazwie niż lokalne konto użytkownika należy adres komputera poprzedzić nazwą użytkownika z dodanym znakiem `@`, np. `kostek@panamint.kcir.pwr.edu.pl`. W przypadku logowania się z lokalnej sieci komputerowej w której znajdują się komputery panamint/diablo można pominąć nazwę domeny (`ssh panamint`).

```
krecik@limba:~:)ssh diablo.kcir.pwr.edu.pl -X
Last login: Fri Oct 18 08:53:06 2024 from 156.17.9.3
------------------------------------------------------------------------------
Oracle Corporation      SunOS 5.10      Generic Patch      SUNW,Sun-Fire-880
******************************************************************************
*                                                                            *
*                               D I A B L O                                  *
*                                 Solaris                                    *
*                                                                            *
*                             przegladarki www: w3m,lynx,firefox             *
*                             klienty pocztowe: alpine,mutt                  *
*                               edytory tekstu: vi,vim,nano,emacs,gedit      *
*                    narzedzia programistyczne: cc,gcc,git,svn               *
* narzedzie do tworzenia dokumentow tekstowych: latex                        *
*                                 czytniki pdf: evince,acroread,xpdf         *
*                            programy graficzne: gimp,dia                    *
*                                                                            *
*             W przypadku problemow prosimy pisac na adres:                  *
*                          admin@kcir.pwr.edu.pl                             *
*                  Aktualnosci: http://diablo.kcir.pwr.edu.pl                *
******************************************************************************
UWAGA!
System jest po nowej instalacji:
-brakujace programy zglaszajcie mailowo lub na OFFTOPie 2020,
-wciaz jeszcze moga wystapic przerwy w dzialaniu serwera.
krecik@diablo:~:
```

Wydruk 2: Terminal użytkownika zalogowanego na diablo

Teraz należy pobrać ze strony kursu plik z programem `trojmian.c`, który posłuży do pracy. Najprościej zrobić to za pomocą polecenia `wget`, wydanego na panamincie lub diablo, mianowicie

```
krecik@panamint:~:wget https://kcir.pwr.edu.pl/~krecik/PProg/Pomoce/trojmian.c
--2024-10-18 13:44:57--  https://kcir.pwr.edu.pl/~krecik/PProg/Pomoce/trojmian.c
Resolving kcir.pwr.edu.pl (kcir.pwr.edu.pl)... 156.17.9.3
Connecting to kcir.pwr.edu.pl (kcir.pwr.edu.pl)|156.17.9.3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1121 (1,1K) [text/x-csrc]
Saving to: ‘trojmian.c’

trojmian.c   100%[=============================================>]   1,09K  --.-KB/s in 0s

2024-10-18 13:44:57 (20,3 MB/s) - ‘trojmian.c’ saved [1121/1121]
```

W ten sposób do lokalnego katalogu zostanie skopiowany plik `trojmian.c`[2] zawierający program na rozwiązywanie równań kwadratowych, pokazany na wydruku 3. Przed przystąpieniem do kompilacji, celem nanoszenia poprawek otworzymy pobrany plik w ulubionym edytorze, np.

```
krecik@panamint:~:emacs trojmian.c &
```

Edytor warto uruchomić w tle, co powoduje dodany na końcu linii znak `&` i pozwoli na wydawanie dalszych poleceń w terminalu bez konieczności zakończenia pracy edytora.

3. Kompilacja i uruchomienie programu

Do błędów, które może zawierać program zaliczamy pojawiające się na etapie kompilacji błędy składniowe (syntaktyczne, zwane też błędami kompilacji), zgłaszane przez konsolidator błędy linkowania, oraz objawiające się

[2]Ponieważ diablo i panamint mają wspólny system plików będzie on dostępny na obu serwerach.

```c
/*
    kompilacja: Sun C: cc -xc99 -Xc trojmian.c -lm
                GNU C: gcc -std=c99 -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>

int main() {
  float a,b,c,delta;

  printf("Program oblicza pierwiastki rownania w postaci\n");
  printf("     2\n");
  printf("  a x + b x +c =0\n");
  print("Podaj wartosc a:");
  scanf("%f",&a);
  printf("Podaj wartosc b:");
  scanf("%f",&b);
  printf("Podaj wartosc c:");
  scanf("%f",&c);

  if (a=0.0)    /*przypadek rownania liniowego */
    if (b!=0.0)
      printf("Jest to równanie liniowe o rozwiazaniu x=%f\n",-c/b)
    else  if (c==0.0)  /* oraz a==b==0.0 */
      printf("Rozwiazaniem jest dowolne x\n");
    else              /* a==b==0.0 != c  */
      printf("Brak rozwiazan\n");
  else {   /*przypadek rownania kwadratowego */
    delta=pow(b,2)-4.0*a*c;
    if (delta<0)
      printf("Brak rozwiazan rzeczywistych\n");
    else            /* delta>=0 */
      if (delta>0)
        printf("Rozwiazaniem sa x1=%f i x2=%f\n",(-b-sqrt(delta))/(2*a),(-b+sqrt(delta))/(2*a));
      else
        printf("Rozwiazaniem sa x1=x2=%f\n",-b/2*a);
  }

}
```

Wydruk 3: Program `trojmian.c`

na etapie wykonywania programu błędy wykonania (powodujące zakończenie działania programu przed zrealizowaniem zadania), błędy logiczne (w efekcie których program zwraca niepoprawne wyniki) i błędy numeryczne (kiedy to program zwraca niedokładne wyniki).

W pliku `trojmian.c` w początkowym komentarzu podane są polecenia, których należy użyć w celu skompilowania programu, jednakże tu rozpoczniemy proces kompilacji od wywołania kompilatorów bez wskazanych opcji, by zobaczyć jakie to powoduje „konsekwencje". Celem będzie usunięcie z programu wszystkich błędów tak, by działał poprawnie dla dowolnych danych wejściowych. Zalecamy realizację całego procesu krok po kroku we wskazanej kolejności, celem poznania wszystkich rodzajów błędów.

3.1 Błędy składniowe

Dla porównania sposobu komunikowania błędów i ostrzeżeń przez różne kompilatory kompilację programu będziemy realizowali symultanicznie na panamincie z wykorzystaniem kompilatora GNU C i na diablo z wykorzystaniem kompilatora Sun C. Jak wspomnieliśmy powyżej cały proces rozpoczniemy od wywołania obu kompilatorów bez

żadnych opcji[3], co daje

```
krecik@panamint:~:gcc trojmian.c
trojmian.c: In function ‘main’:
trojmian.c:13:3: warning: implicit declaration of function ‘print’; did you mean
‘lrint’? [-Wimplicit-function-declaration]
   13 |   print("Podaj wartosc a:");
      |   ^~~~~
      |   lrint
trojmian.c:22:67: error: expected ‘;’ before ‘else’
   22 |       printf("Jest to równanie liniowe o rozwiazaniu x=%f\n",-c/b)
      |                                                                   ^
      |                                                                   ;
   23 |     else  if (c==0.0)  /* oraz a==b==0.0 */
      |     ~~~~
```

```
krecik@diablo:~:cc trojmian.c
"trojmian.c", line 13: warning: implicit function declaration: print
"trojmian.c", line 23: syntax error before or at: else
cc: acomp failed for trojmian.c
```

Jak widać kompilator na diablo jest bardziej lakoniczny, ale oba informują o błędzie składniowym przed znajdującym się w linii 23 słowem kluczowym `else` i ostrzegają przed niejawną definicją funkcji `print`. Na razie zignorujemy ostrzeżenie i usuniemy jedynie błąd przez dodanie średnika na końcu linii 22. Kolejna próba kompilacji daje w efekcie

```
krecik@panamint:~:gcc trojmian.c
trojmian.c: In function ‘main’:
trojmian.c:13:3: warning: implicit declaration of function ‘print’; did you mean
‘lrint’? [-Wimplicit-function-declaration]
   13 |   print("Podaj wartosc a:");
      |   ^~~~~
      |   lrint
/usr/bin/ld: /tmp/ccvW69ns.o: in function `main':
trojmian.c:(.text+0x45): undefined reference to `print'
/usr/bin/ld: trojmian.c:(.text+0x1a4): undefined reference to `pow'
/usr/bin/ld: trojmian.c:(.text+0x24a): undefined reference to `sqrt'
/usr/bin/ld: trojmian.c:(.text+0x29e): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
```

```
krecik@diablo:~:cc trojmian.c
"trojmian.c", line 13: warning: implicit function declaration: print
Undefined                       first referenced
 symbol                             in file
pow                                 trojmian.o
sqrt                                trojmian.o
print                               trojmian.o
ld: fatal: symbol referencing errors. No output written to a.out
```

Teraz poniżej zignorowanego wcześniej ostrzeżenia pojawiają się nowe komunikaty – są one generowane przez linker, który został automatycznie uruchomiony po tym, gdy swoje działanie zakończył kompilator[4].

[3] Osoby bardziej dociekliwe mogą dodatkowo realizować zadanie z kompilatorem GNU C na diablo, który jest tam w starszej wersji – polecenie `gcc trojmian.c`.

[4] Dzieje się tak, gdyż polecenia `gcc/cc` uruchamiają nie tylko kompilator, ale także inne programy używane w procesie kompilacji, w tym linker (polecenie `ld`). De facto `gcc` jest także kompilatorem języka C++. By nie uruchamiać linkera po poprawnie zakończonej kompilacji należy użyć opcji `-c`.

3.2 Błędy linkowania

Linker narzeka na pojawiające się niezdefiniowane symbole: `pow`, `sqrt` oraz zgłaszany wcześniej przez kompilator `print`. Dzieje się tak, gdyż symbole te nie są zdefiniowane ani w kompilowanym programie ani w dołączanych automatycznie bibliotekach systemowych. I faktycznie, w bibliotekach systemowych nie ma definicji funkcji `print` i łatwo zauważyć, że w programie jest błąd literowy (instrukcja `print` w miejsce `printf` – jak widać sugestia używanego na panamincie kompilatora GNU C w postaci ostrzeżenia „warning: implicit declaration of function 'print'; did you mean 'lrint'?" nie była poprawna), zaś symbole `pow` i `sqrt` są zdefiniowane w bibliotece matematycznej, która nie jest dołączana do programu automatycznie. By dołączyć wybraną bibliotekę należy użyć opcji `-l` z podaną nazwą biblioteki, w tym wypadku `m`. Po dopisaniu w pliku brakującego 'f' i przy życiu opcji `-l` otrzymujemy

```
krecik@panamint:~:gcc trojmian.c -lm
krecik@diablo:~:cc trojmian.c -lm
```

co oznacza, że proces tworzenia wersji wykonywalnej programu zakończył się sukcesem i znajduje się ona w pliku o nazwie `a.out`.

3.3 Błędy wykonania programu

Teraz możemy przejść do uruchomienia programu. Wywołanie polecenia `./a.out`[5] daje

```
krecik@panamint:~:./a.out
-bash: ./a.out: cannot execute binary file: Exec format error
```

co oznacza, że program nie może zostać wykonany ze względu na błąd formatu. Dzieje się tak, gdyż ostatni proces kompilacji uruchomiliśmy na serwerze diablo, co spowodowało, że w pliku `a.out` umieszczona została wersja wykonywalna programu dla architektury komputera Sun Fire V880 wykorzystującego procesory UltraSPARC III z zestawem instrukcji SPARC[6], który nie jest kompatybilny z zestawem instrukcji x86-64[7] znajdujących się w panamincie procesorów AMD Opteron 6276. Zagadnienie to jest związane z przenośnością oprogramowania: programy wykonywalne nie są „przenośne" pomiędzy różnymi architekturami a programy napisane w językach wysokiego poziomu (jakim jest język C) mogą być przenośne[8]. I faktycznie, wydanie polecenia `a.out` na diablo daje

```
krecik@diablo:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:2
Podaj wartosc c:-8
Rozwiazaniem sa x1=-Inf i x2=NaN
```

Program został uruchomiony i po podaniu wartości współczynników 1, 2 i -8 zwrócił wynik w postaci dwóch symboli: `-Inf` i `NaN`[9] będących efektem błędu wykonania programu, polegającego na wywołaniu instrukcji dzielenia przez zero. Uruchomienie programu dla innych wartości współczynników pokazuje, że są to najczęściej zwracane wartości, choć czasami (np. dla 1, 0, -4) uzyskujemy

```
krecik@diablo:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:-4
Rozwiazaniem sa x1=x2=-0.000000
```

[5] Początkowe `./` wskazuje, że uruchamiany plik znajduje się w katalogu bieżącym.
[6] który należy do klasy „zredukowanego zestawu instrukcji" RISC
[7] znanym także pod nazwami x64, x86_64, AMD64 i Intel 64, który należy do klasy „rozbudowanego zestawu instrukcji" CISC
[8] Pod warunkiem, że spełniają wymagania standardu języka C i nie wykorzystują bibliotek niestandardowych.
[9] oznaczających „-Infinitty" i „Not a Number"

co ewidentnie nie jest poprawnym wynikiem. Oznacza to, że w programie znajdują się powodujące takie działanie błędy logiczne, którymi zajmiemy się w dalszej kolejności.

Ponieważ przedstawiliśmy już pokrótce zagadnienie przenośności oprogramowanie, dla ustalenia uwagi będziemy teraz korzystali jedynie z kompilatora GNU C na panamincie. I tak, by uzyskać więcej informacji o procesie kompilacji wykorzystamy opcje `-pedantic` i `-Wall`[10], które powodują, że kompilator staje się „bardziej rozgadany"

```
krecik@panamint:~:gcc -pedantic -Wall trojmian.c -lm
trojmian.c: In function ‘main’:
trojmian.c:20:7: warning: suggest parentheses around assignment used as truth
value [-Wparentheses]
   20 |   if (a=0.0)    /*przypadek rownania liniowego */
      |       ^
```

Widzimy, że kompilator zwrócił naszą uwagę na linię 20 programu, w której sugeruje użycie dodatkowych nawiasów wokół instrukcji `a=0.0`. Możemy oczywiście te nawiasy dodać, co „uciszy" kompilator, ale nie zapewni usunięcia błędu z programu[11]. W tym przypadku kluczowym słowem w ostrzeżeniu jest słowo „przypisanie" – faktycznie w programie użyta została instrukcja przypisania (`=`) w miejsce instrukcji porównania (`==`), co jest błędem logicznym powodującym błąd wykonania programu[12]. Po poprawieniu tego błędu dostajemy

```
krecik@panamint:~:gcc -pedantic -Wall trojmian.c -lm
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:2
Podaj wartosc c:-8
Rozwiazaniem sa x1=-4.000000 i x2=2.000000
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:-4
Rozwiazaniem sa x1=-2.000000 i x2=2.000000
```

Widać, że teraz dla wcześniej analizowanych przypadków program działa poprawnie (i nie chodzi tylko o to, że zwraca dwa pierwiastki, ale że pierwiastki te są **faktycznie** rozwiązaniem równania :).

3.4 Inne błędy logiczne

Naszym zadaniem jest teraz sprawdzenie czy program nie zawiera innych błędów logicznych i czy działa poprawnie dla wszystkich wartości danych wejściowych. W tym celu musimy po pierwsze zapewnić, że przetestujemy wszystkie możliwe rozgałęzienia algorytmu realizowanego przez program (równanie kwadratowe dwa pierwiastki rzeczywiste, jeden pierwiastek podwójny, brak pierwiastków rzeczywistych, równanie liniowe, sprzeczne, tożsamościowe). Najlepiej do testów użyć takich wartości współczynników, dla których znamy rozwiązanie, więc na przykład dla $(x-3)(x+1.5) = x^2 - 1.5x - 4.5 = 0$, $(x-3)^2 = x^2 - 6x + 9 = 0$, $x^2 + 4 = 0$ oraz $2x - 3 = 0$, $3 = 0$ i $0 = 0$. Tak więc

[10] Warto dodać, że kompilator Sun C nie ma tych opcji, co gorzej, nie ma opcji im równoważnych.

[11] Przykład ten pokazuje, by nie postępować według sugestii kompilatora bez zrozumienia ich zasadności – takie działanie może nam jedynie utrudnić znalezienie i usunięcie rzeczywistych błędów.

[12] W efekcie tego błędu, jakiejkolwiek wartości nie podalibyśmy jako wartość zmiennej a, w linii 20 programu przyjmie ona wartość 0, która zinterpretowana jako fałsz powoduje przejście programu do rozwiązywania przypadku równania kwadratowego, w którym pojawia się dzielenie przez a (zastanawiające jedynie, dlaczego nie pojawiło się dzielenie przez 0 gdy jako współczynniki równania podaliśmy wartości 1, 0 i -4, ale nie uprzedzajmy faktów :).

```
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:-1.5
Podaj wartosc c:-4.5
Rozwiazaniem sa x1=-1.500000 i x2=3.000000
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:-6
Podaj wartosc c:9
Rozwiazaniem sa x1=x2=3.000000
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:4
Brak rozwiazan rzeczywistych
```

```
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:0
Podaj wartosc b:2
Podaj wartosc c:-3
Jest to równanie liniowe o rozwiazaniu x=1.500000
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:0
Podaj wartosc b:0
Podaj wartosc c:3
Brak rozwiazan
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:0
Podaj wartosc b:0
Podaj wartosc c:0
Rozwiazaniem jest dowolne x
```

Wszystkie uzyskane rozwiązania są poprawne jednakże zauważmy, że wszystkie nasze dane testowe dla przypadku równania kwadratowego były „tendencyjne": jako wartość współczynnika a zawsze podawaliśmy 1. Spróbujmy więc przetestować program dla a różnych od 1 - wiemy, że pomnożenie równania obustronnie przez jakąś niezerową wartość nie zmienia jego pierwiastków, weźmy więc dwa pierwsze równania z poprzedniego testu i pomnóżmy np. przez 2: $2(x-3)(x+1.5) = 2x^2 - 3x - 9 = 0$, $2(x-3)^2 = 2x^2 - 12x + 18 = 0$ otrzymując

```
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:2
Podaj wartosc b:-3
Podaj wartosc c:-9
Rozwiazaniem sa x1=-1.500000 i x2=3.000000
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:2
Podaj wartosc b:-12
Podaj wartosc c:18
Rozwiazaniem sa x1=x2=12.000000
```

Widzimy, że podczas gdy w przypadku równania z dwoma różnymi pierwiastkami nie zmieniły się one, to w przypadku równania z pierwiastkiem podwójnym wzrósł on czterokrotnie. Spojrzenie na linię 35 w której wyliczana jest wartość pierwiastka podwójnego pozwala stwierdzić, że brakuje w niej nawiasów wokół wyrażenia `2*a` co powoduje, że wynik zamiast być dzielony przez wartość a zostaje przez nią pomnożony (stąd czterokrotny wzrost pierwiastka przy a równym 2)[13]. Dopisujemy nawiasy, kompilujemy i powtarzamy test

```
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:2
Podaj wartosc b:-12
Podaj wartosc c:18
Rozwiazaniem sa x1=x2=3.000000
```

otrzymując poprawny wynik.

Przy usuwaniu błędów warto dbać, by nie wprowadzać nowych błędów (łatwo powiedzieć!) oraz by program spełniał wymagania wybranej normy języka C wywołując kompilator z odpowiednimi opcjami. Dla zapewnienia przestrzegania przyjętej na naszych zajęciach normy C99 (ISO/IEC 9899:1999) i rozszerzonego poziomu wypisywanych przez kompilator komunikatów zalecamy wywoływanie kompilatorów z opcjami jak poniżej

```
krecik@panamint:~:gcc -std=c99 -pedantic -Wall trojmian.c -lm
```

```
krecik@diablo:~:cc -xc99 -Xc trojmian.c -lm
```

By zobaczyć jakie inne opcje można podać w wywołaniu kompilatora wystarczy wywołać polecenie (sekcja OPTIONS :)

```
krecik@panamint:~:man gcc
```

lub

```
krecik@diablo:~:man cc
```

[13] I stąd brak dzielenia przez zero w tym przypadku, gdy w programie zamiast operacji porównania zmiennej a z zerem była użyta operacja przypisania.

3.4 Błędy numeryczne

Sprawdźmy teraz jak nasz program działa dla „dużych” liczb. W tym celu rozwiążemy równania postaci $x^2 = d^2$, gdzie d jest stałą równą kolejno 10^3, 10^6, 10^{10} i 10^{20}. Wpisujemy więc

```
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:-1000000
Rozwiazaniem sa x1=-1000.000000 i x2=1000.000000
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:-1000000000000
Rozwiazaniem sa x1=-999999.997952 i x2=999999.997952
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:-10000000000000000000
Rozwiazaniem sa x1=-10000000100.204386 i x2=10000000100.204386
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:1
Podaj wartosc b:0
Podaj wartosc c:-100000000000000000000000000000000000000000
Rozwiazaniem sa x1=-inf i x2=inf
```

Widzimy, że jedynie w pierwszym przypadku program zwrócił „dokładny” wynik – w pozostałych widać ewidentne błędy numeryczne lub nawet wartości reprezentujące nieskończoność. W przypadku realizacji programów komputerowych (szczególnie służących symulacji zachowania układów dynamicznych) nie wolno o tego typu błędach zapominać.

Na koniec zapytajmy jeszcze, jak będzie działał nasz program, gdy na jego wejście nie podamy wartości numerycznych. Uruchamiamy program i jako pierwszą wartość wpisujemy napis `ala` otrzymując

```
krecik@panamint:~:./a.out
Program oblicza pierwiastki rownania w postaci
     2
  a x + b x +c =0
Podaj wartosc a:ala
Podaj wartosc b:Podaj wartosc c:Rozwiazaniem jest dowolne x
```

Ale to już temat na zupełnie inną historię.