Anna Jasielec

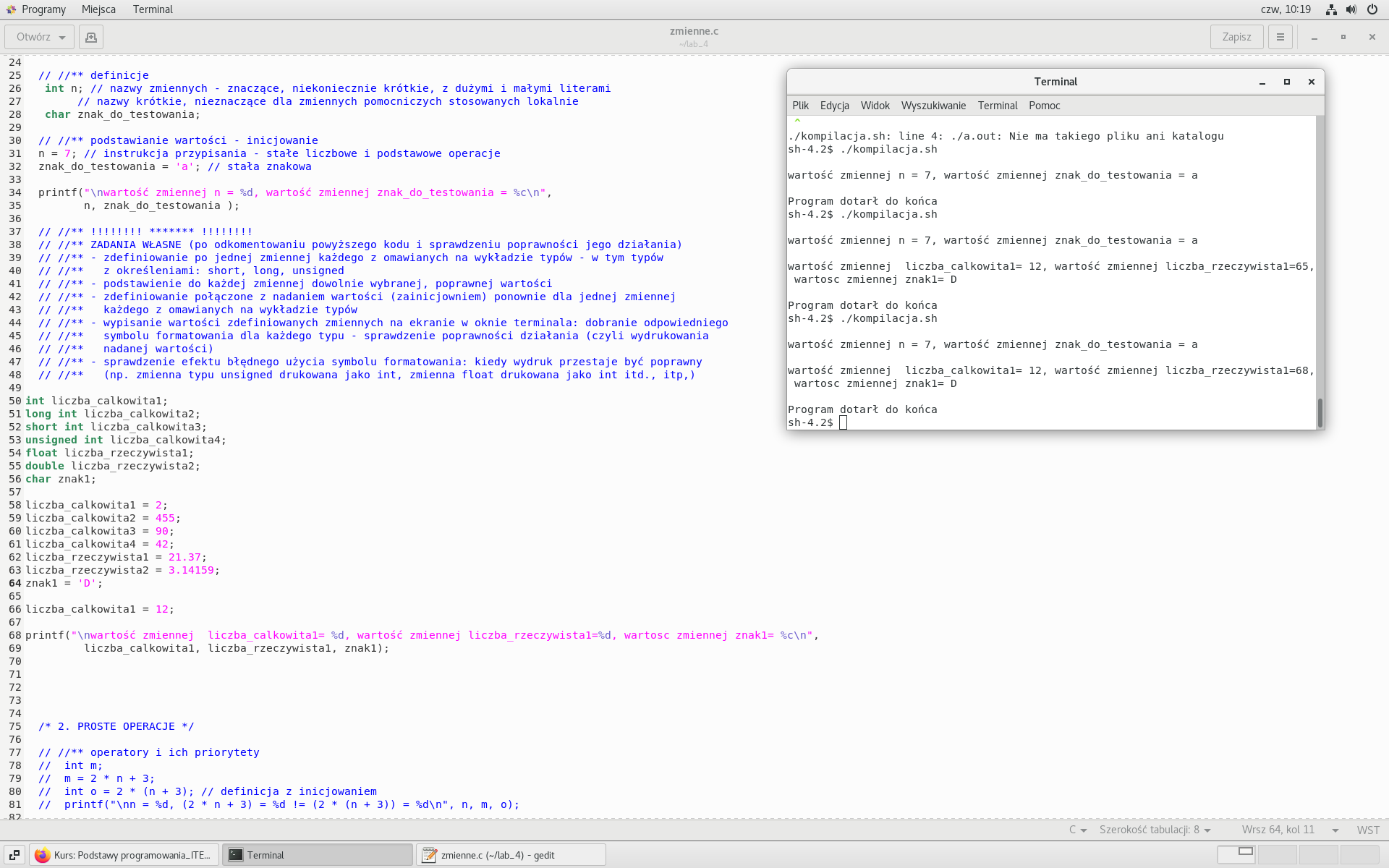
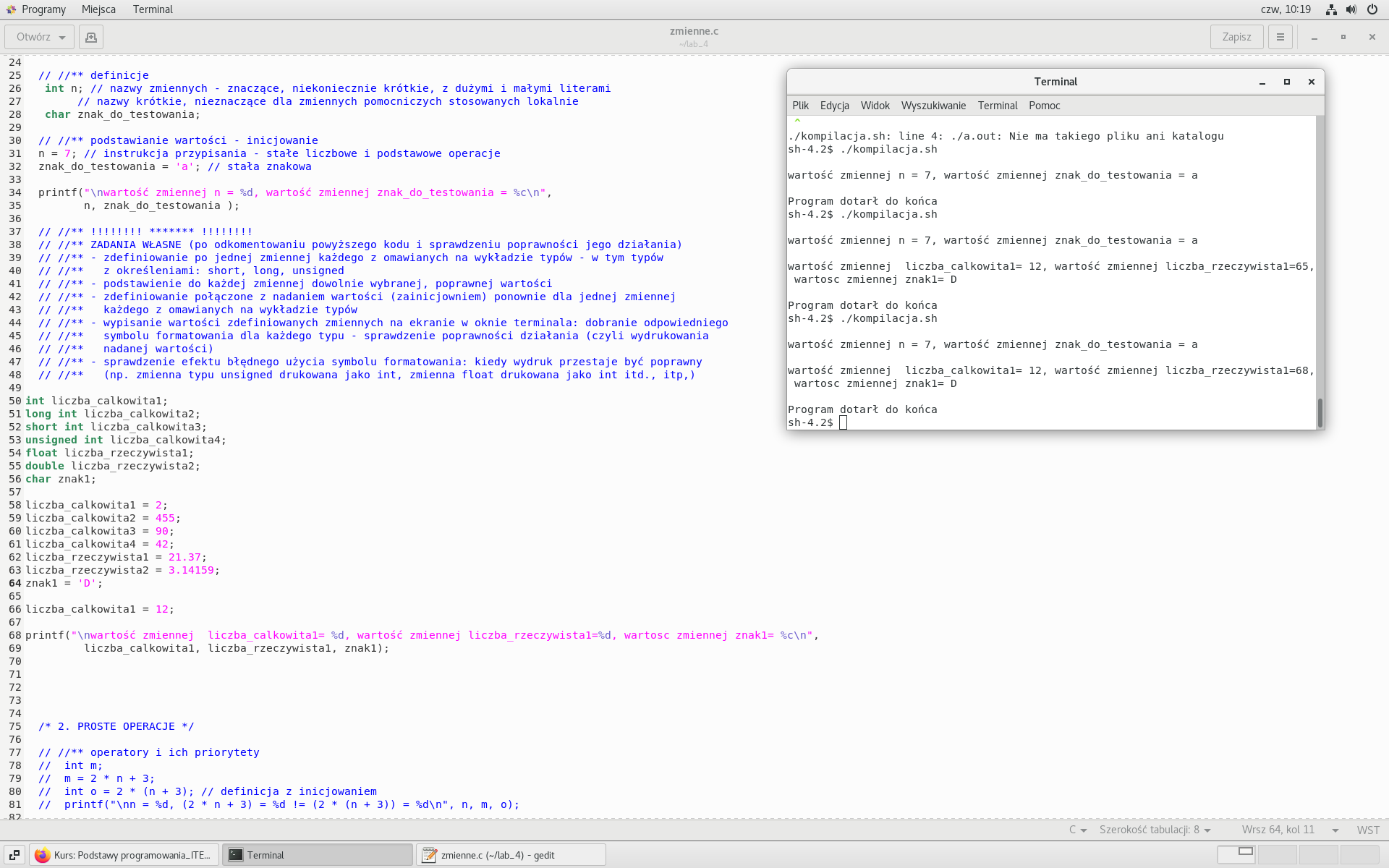
Podstawy programowania, grupa nr 4

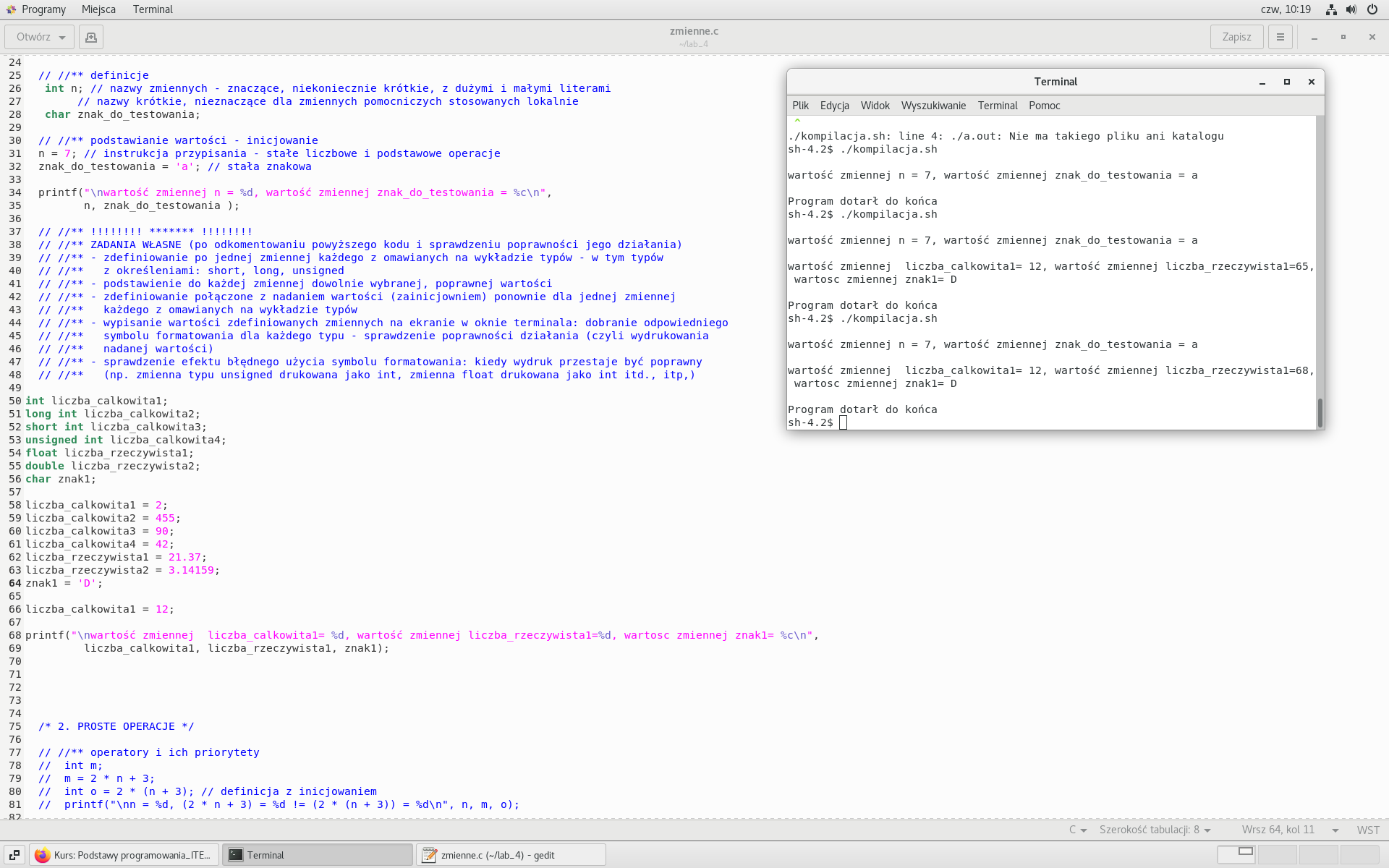
Sprawozdanie z laboratorium nr 4

**Cel laboratorium:** Opanowanie tworzenia prostych programów w C zapisujących wartości zmiennych różnych typów oraz wykonujących podstawowe operacje na zmiennych.

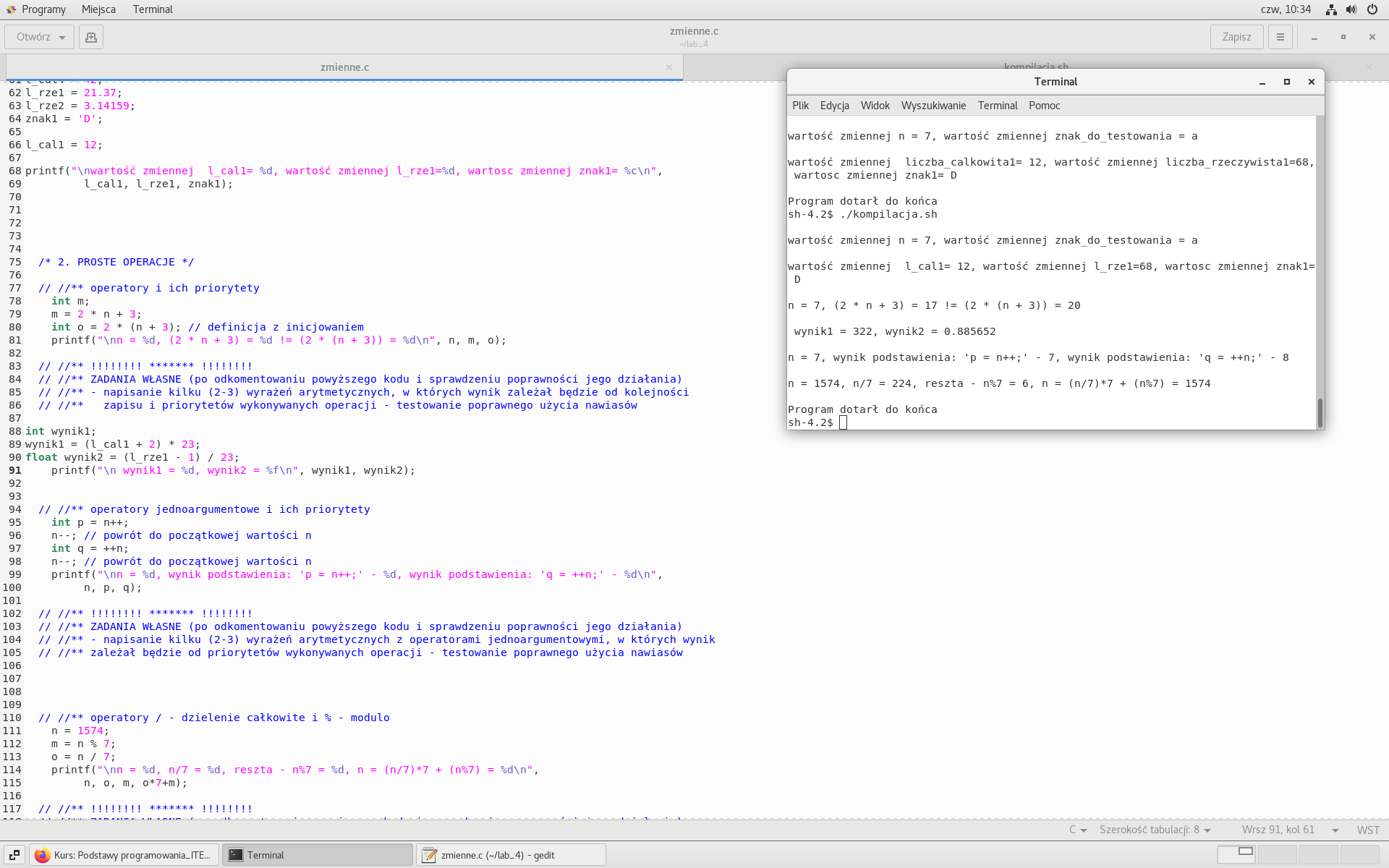
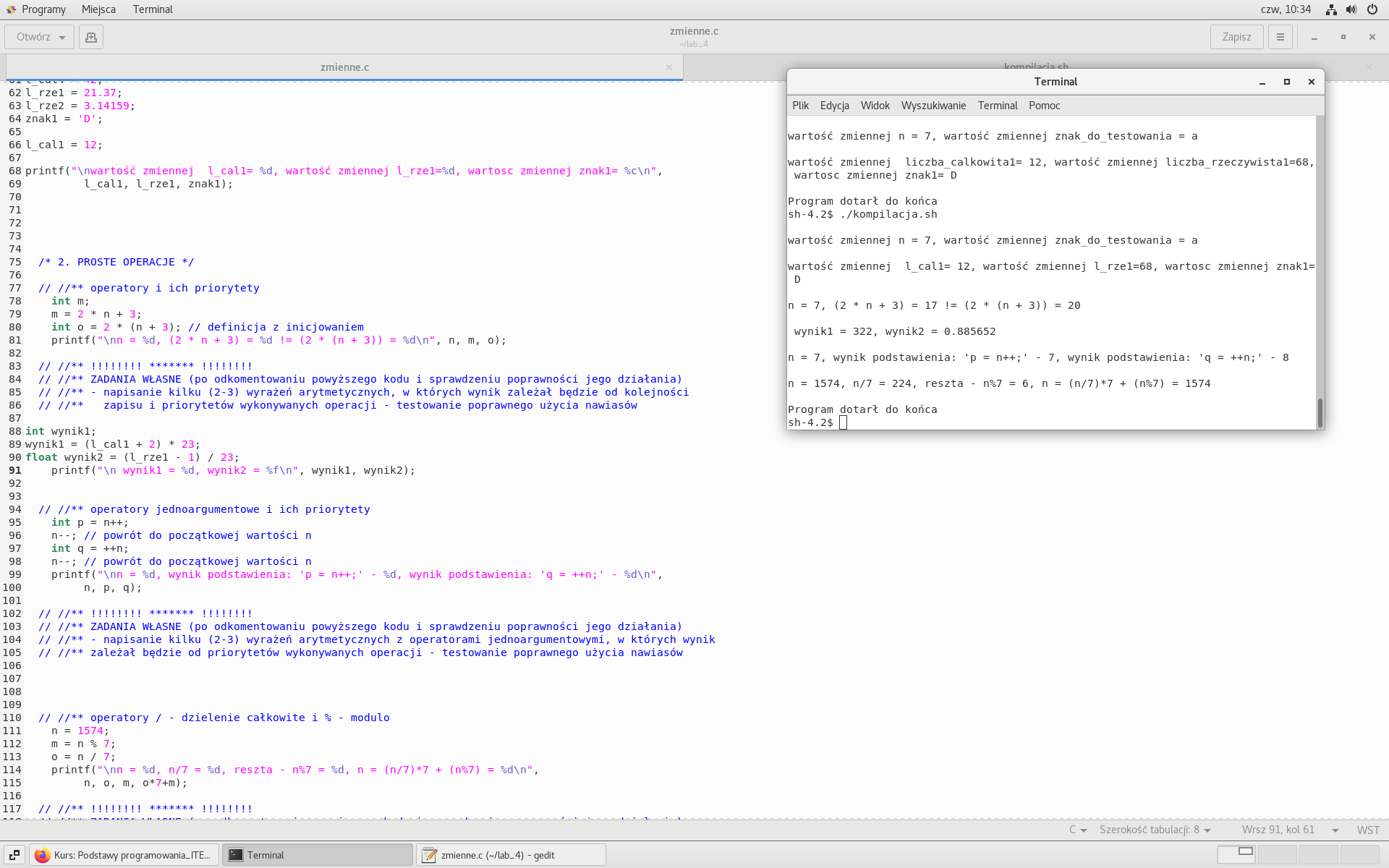
**Przebieg zajęć:**

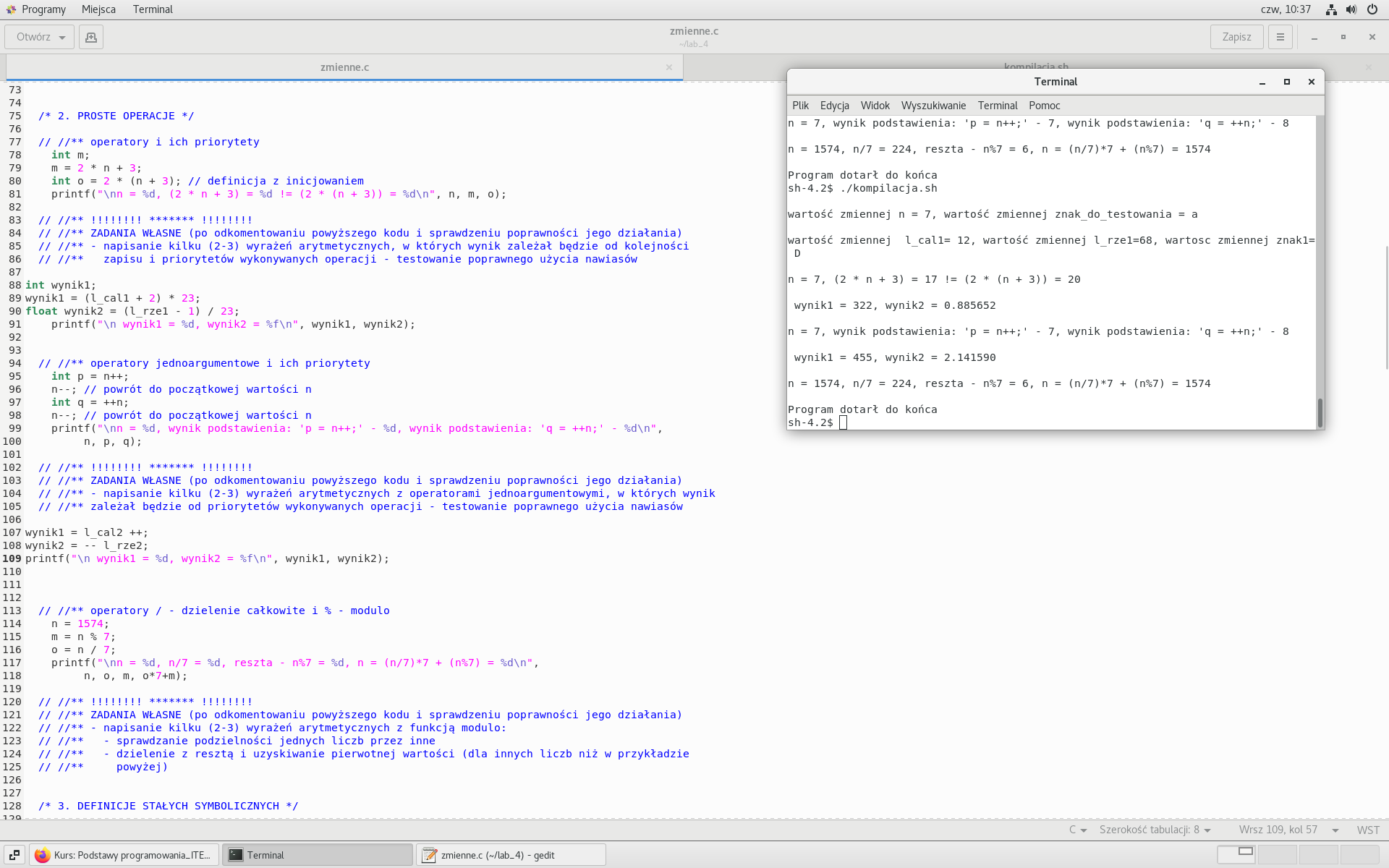
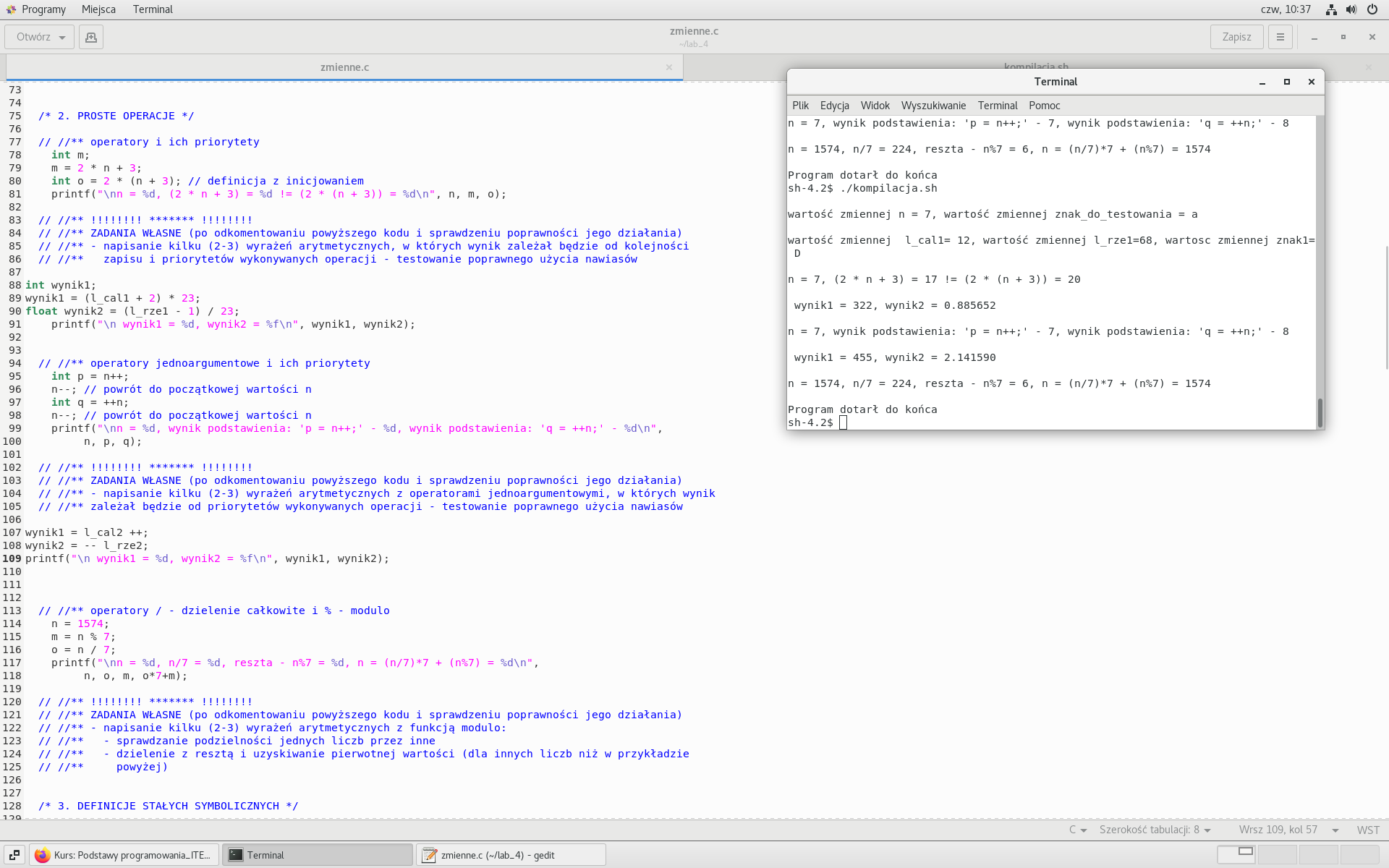
* otworzenie programu zmienne.c,
* kompilacja i sprawdzanie działanie programu,
* odkomentywywanie, analiza przykładów i wykonywanie zadań:

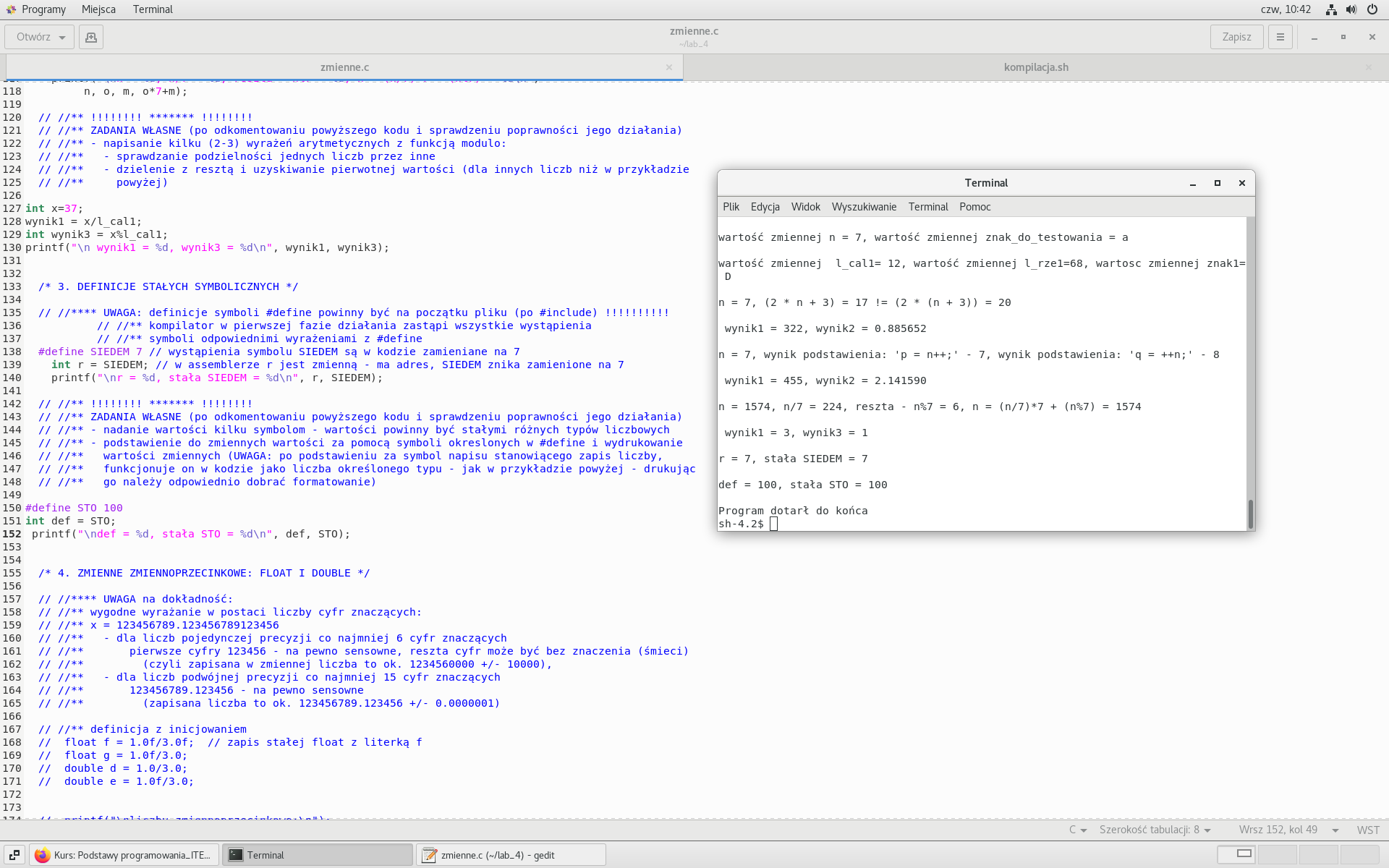
**Zadanie 1.** Definiowanie zmiennych, inicjowanie i wypisywanie ich wartości

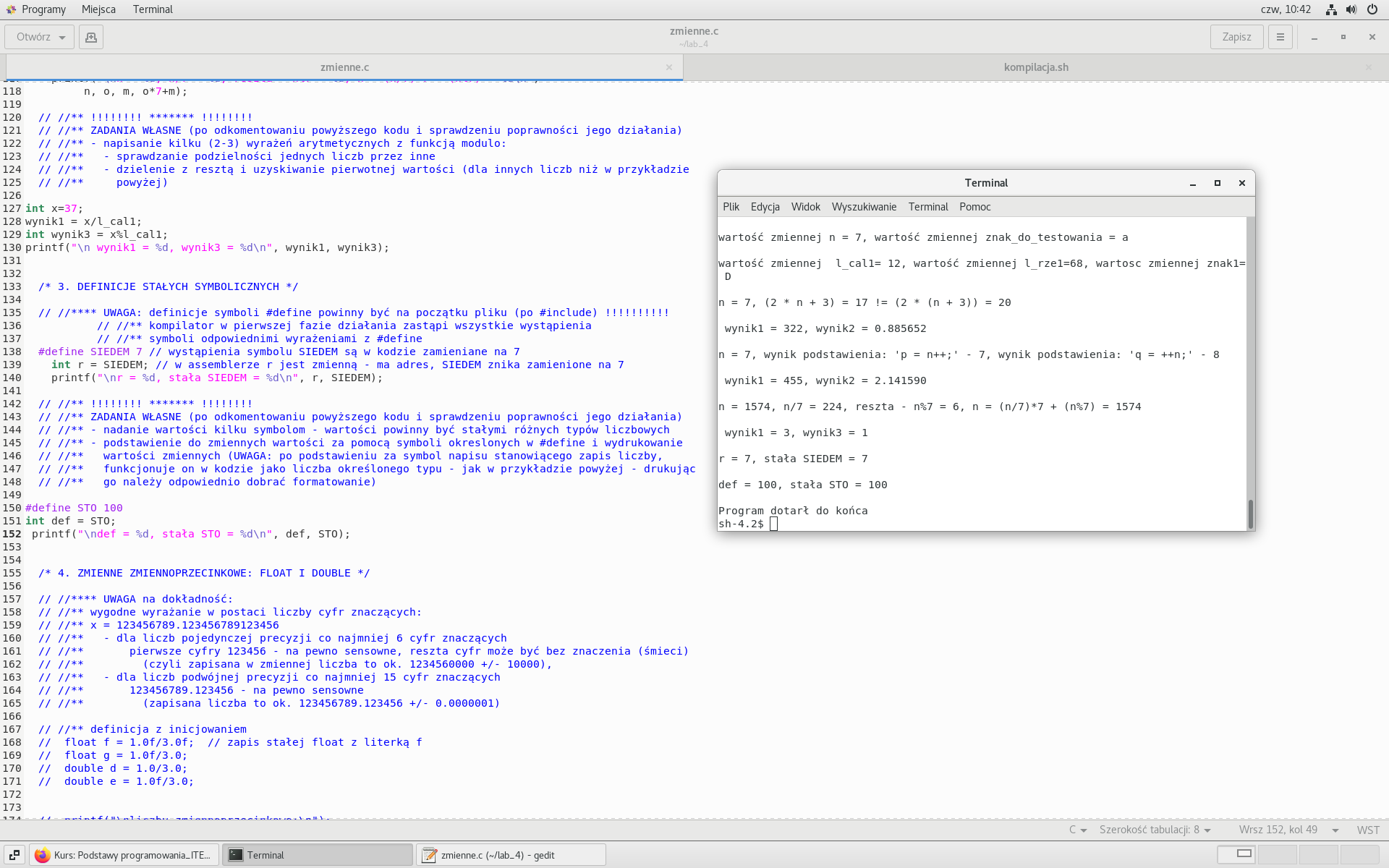


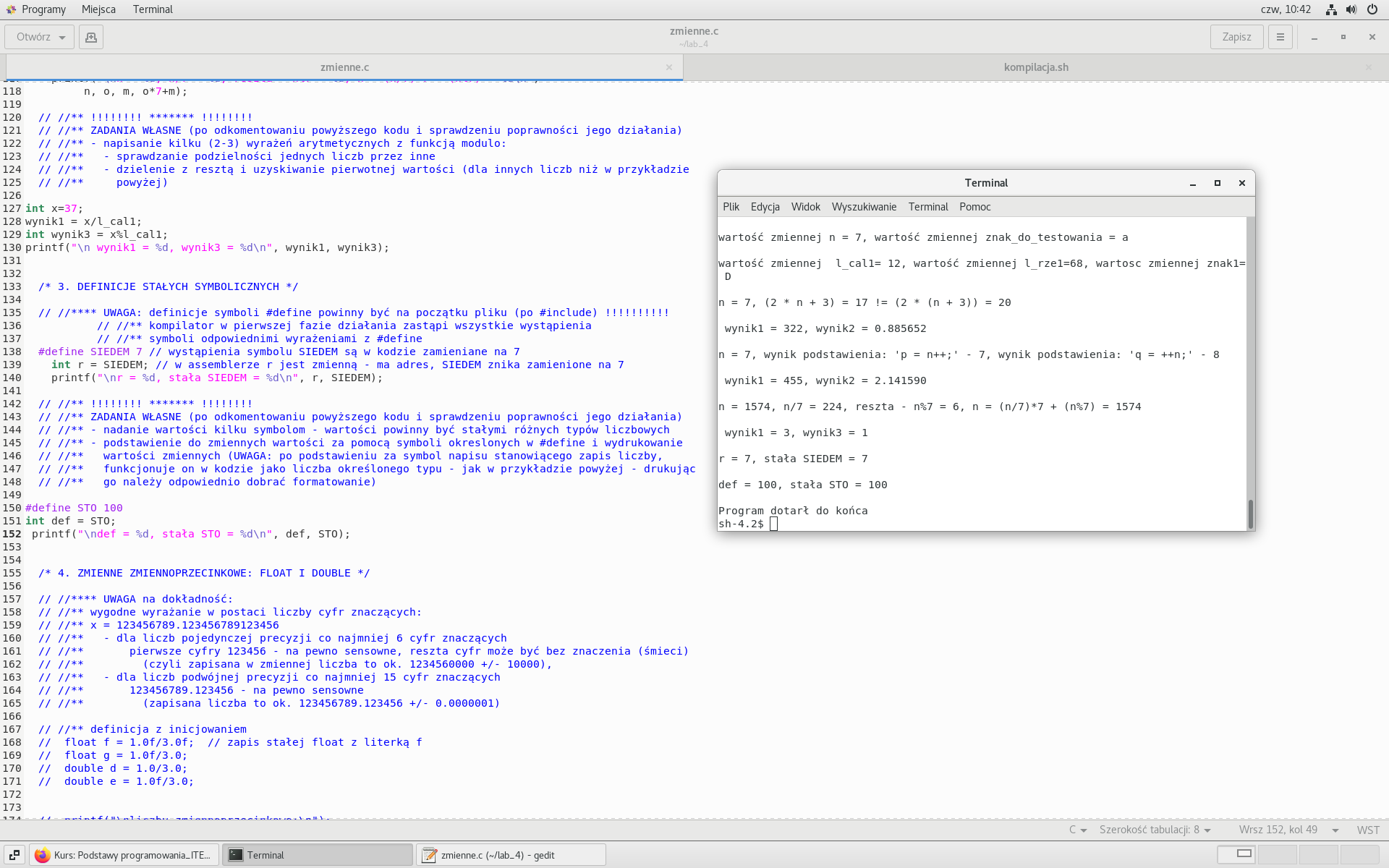
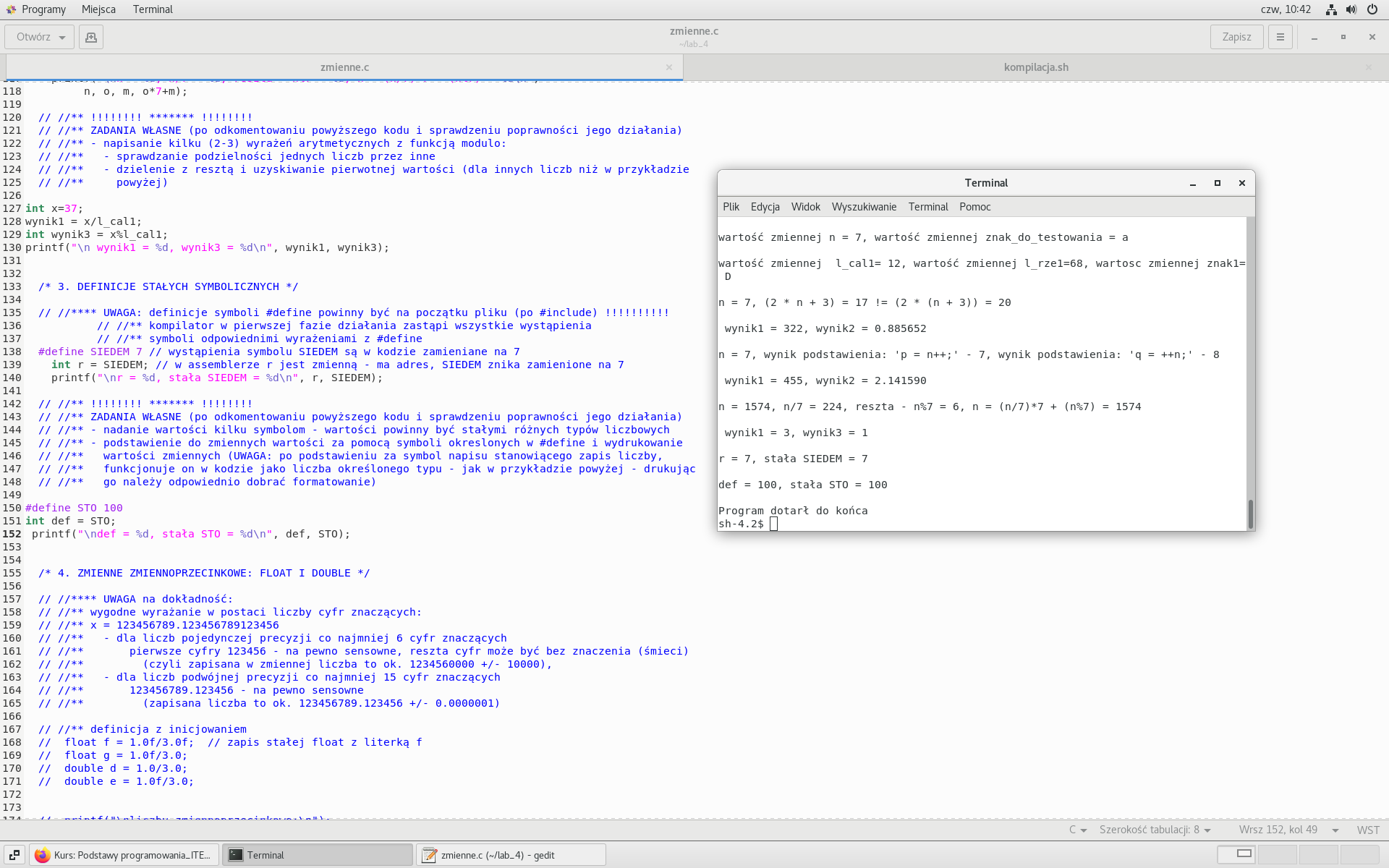
**Zadanie 2.** Proste operacje na zmiennych.  
- sprawdzenie zachowania kolejności wykonywania działań,

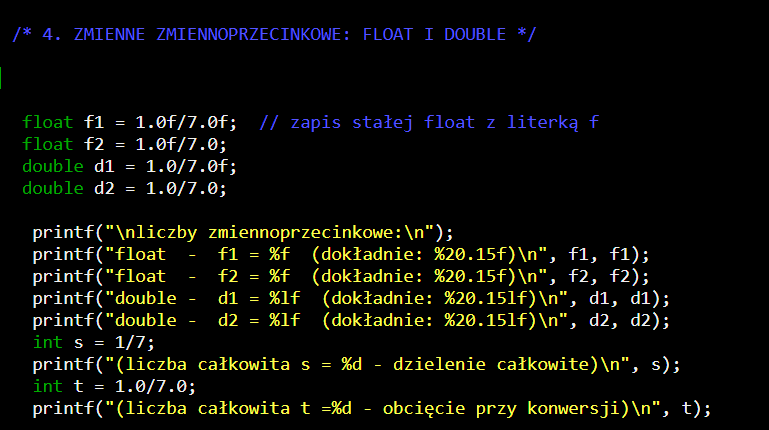


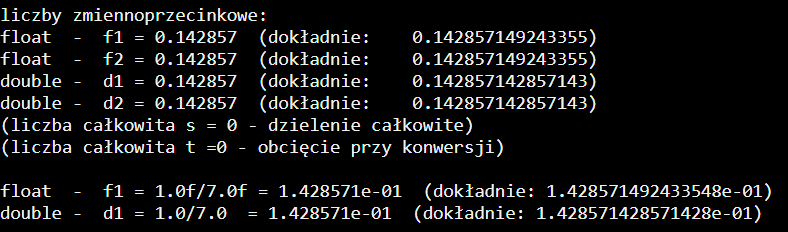
- inkrementacja, dekrementacja, preinkrementacja, predekrementacja,

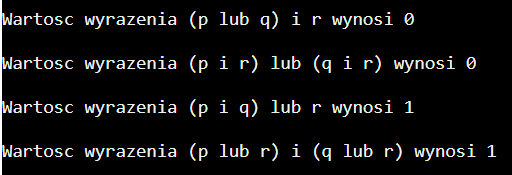
- dzielenie całkowite i modulo(%),

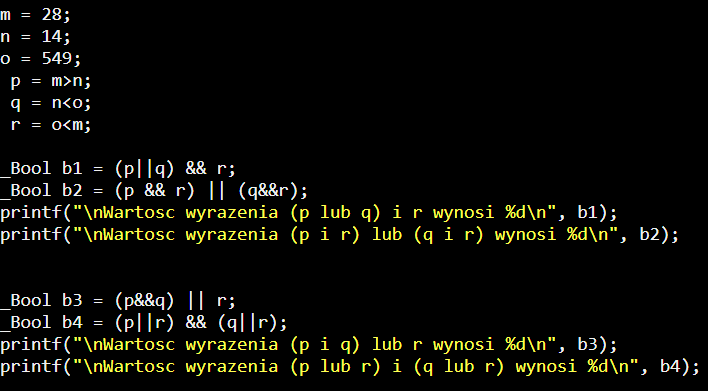


**Zadanie 3.** Definiowanie stałych symbolicznych.

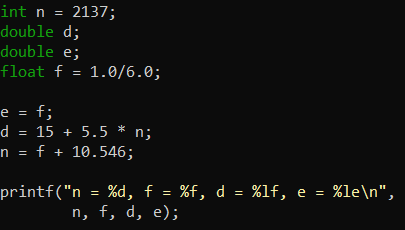
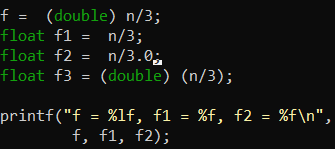
**Zadanie 4.** Definiowanie z float i double, sprawdzanie dokładności matematycznej wartości.



**Zadanie 5.** Konstruowanie złożonych wyrażeń logicznych.

****

- (p lub q) i r =? (p i r) lub (q i r) jest równoważne, (p i q) lub r =? (p lub r) i (q lub r) jest równoważne,  
- w przypadku usuwania kolejnych nawiasów kolejność jest od lewej do prawej, wynik się zmienia.

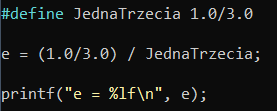
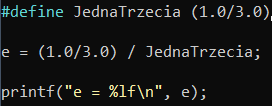
**Zadanie 6.** Niejawne i jawne konwersje typów.



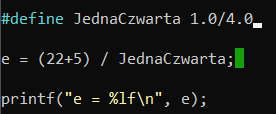
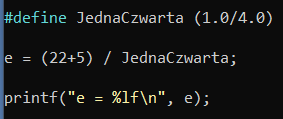


- kompilator niejawnie skonwertował wartości zmiennych na inne typy.

**Zadanie 7.** Definiowanie ułamków, sprawdzanie roli nawiasów w definicji.



- gdy nie użyłam nawiasu podczas definiowania stałej JednaTrzecia kompilator wykonywał równanie od lewej do prawej (1/3 podzielił na 1, następnie podzielił na 3), stąd wynik e = 0.111111.

- własne przykłady:





**Zadanie 8.** Analizowanie problemów związanych z precyzją liczb zmiennoprzecinkowych.

- Wynikiem działania (fx1+fx2)-fx1 powinna być zmienna fx2, jednak tak nie jest, ponieważ procesor nie potrafi dokładnie reprezentować takich liczb w systemie binarnym. Dzieje się tak nawet wtedy kiedy nadamy zmiennej typ *double*.

- Liczba przez którą dzielimy jest za blisko 0. Aby naprawić ten błąd należało przyjąć tolerancję 1e-12. Poniżej niej dwie liczby zmiennoprzecinkowe są uznawane za równe.

**Wnioski:**

* *int* definiuje wartość całkowitą, *float* wartość zmiennoprzecinkową, *char(‘ ‘)* pojedynczy znak, *short* lub *long* określa nam zakres wartości jakie zmienna przyjmie, używając *unsigned* zmienna będzie mogła być tylko dodatnia, ale zwiększy się jej dodatni zakres.
* Kompilator kompiluje działania zgodnie z zasadą kolejności wykonywania działań.
* Inicjując zmienną z użyciem inkrementacji najpierw przypiszemy do zmiennej\_a wartość zmiennej\_b, a następnie podniesiemy wartość zmiennej\_b o 1, używając preinkrementacji przypisujemy zmiennej\_a podniesioną o 1 wartość zmiennej\_b. Dekrementacja i predekrementacja działają podobnie, z tą różnicą, że zmiejszamy wartość o 1.
* Gdy dzielimy przez siebie dwie zmienne typu *int* wynik będzie liczbą całkowitą.
* Modulo (%) pokaże nam resztę z dzielenia dwóch liczb całkowitych.
* Gdy użyjemy *#define symbol stała* kompilator zastąpi wszystkie użycia symbolu w programie na określoną stałą. Przy definiowaniu ułamków należy pamiętać o nawiasach.
* Zmienna typu *double* będzie miała większą dokładność niż zmienna typu *float*.
* *\_Bool* to typ zmiennej, który przechowywuje dane prawda(1) i fałsz(0).
* && oznacza i, || oznacza lub, kompilator przy kompilowaniu złożonych wyrażeń logicznych uwzględnia nawiasy.
* Jeżeli przypiszemy do zmiennej\_a pewnego typu, wartość zmiennej\_b innego typu to kompilator niejawnie konwertuje tą wartość na typ zmiennej\_a.
* Chcąc zmienić typ zmiennej możemy użyć konwersji jawnej, czyli: *zmienna = (typ\_zmiennej) wartość\_zmiennej.*