Coś poszło nie tak

#błąd_kompilacji
#błąd_wykonania

Nie zawsze programowanie idzie tak gładko. Nawet doświadczeni programiści popełniają błędy, których wyszukiwanie i usuwanie jest bardzo ważnym elementem warsztatu informatyka.

Chyba najczęstszym błędem popełnianym przez początkujących programistów jest opuszczenie średnika `;` na zakończeniu instrukcji, na przykład:

```
int n, m;
cin >> n
cin >> m
```

Taki błąd jest wykrywany na etapie kompilacji kodu i objawia się komunikatem w rodzaju:

```
error: expected ';' before 'cin'
```

czyli, że spodziewano się średnika przed `cin`. Dodatkowo kompilator podaje numer wiersza, w którym wykryto błąd. W tym przypadku będzie to ostatni wiersz podanego fragmentu kodu – dzieje się tak dlatego, że podział tekstu na linijki jest dość dowolny i teoretycznie instrukcja `cin >> n` mogłaby być zwieńczona średnikiem w następnym wierszu (choć nikt oczywiście tak nie robi).

Innym typowym błędem jest użycie niezadeklarowanej zmiennej lub funkcji, na przykład:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
  cin >> n;
  return 0;
}
```

Wtedy dostaniemy komunikat:

```
error: 'n' was not declared in this scope
```

czyli że zmienna n nie została zadeklarowana w tej części programu, gdzie usiłujemy jej użyć.

Jeśli podczas kompilacji kodu pojawi się błąd, wtedy – rzecz jasna – nie zostanie utworzony plik wykonywalny i nie będziemy mogli uruchomić naszego programu.

Jeśli kompilator sygnalizuje, że niezadeklarowane jest `cin` lub `cout`, wtedy należy sprawdzić początkowe linijki naszego programu (te z komendami `include ...` oraz `using ...`), bo najprawdopodobniej tam właśnie popełniliśmy błąd.

Oprócz komunikatu o błędzie (ang. **error**) możemy napotkać również ostrzeżenie (ang. **warning**, wymawiaj: *łorning*), które nie powoduje przerwania kompilacji. Plik wykonywalny zostanie utworzony i będzie można go uruchomić, ale trzeba liczyć się z tym, że może on działać niezgodnie z naszymi oczekiwaniami.

Typowa sytuacja, gdy pojawia się ostrzeżenie, zachodzi wtedy, gdy zapiszemy jakiś tekst w pojedynczych apostrofach (`' '`) zamiast w cudzysłowie (`" "`), na przykład:

```
string s = 'xyz';
```

Rzecz w tym, że w pojedynczych apostrofach zapisuje się pojedyncze znaki (typ danych `char`), zaś w cudzysłowach zapisujemy ciągi znaków (dokładniej: tablice znaków). Przy takim błędzie możemy spodziewać się komunikatu:

```
warning: multi-character character constant
```

który nie wróży nic dobrego, taki program na bank nie będzie działał prawidłowo, choć pozwoli się „odpalić". Obowiązuje tu zasada: *No news is good news**, do której warto się stosować.

Nie wszystkie błędy wykrywane są przez kompilator – generalnie rzecz biorąc kompilator języka C++ jest dość liberalny i pozwala czasem programiście na karkołomne konstrukcje, zakładając że autor kodu „wie, co robi" – to niekoniecznie musi być prawdą w przypadku początkujących programistów. Dość charakterystyczny błąd niesygnalizowany przez kompilator polega na omyłkowym użyciu pojedynczego znaku równości zamiast podwójnego, na przykład:

```
if(n = 0)
  cout << "Hej!";
```

W powyższym przykładzie wyrażenie znajdujące się w nawiasie będzie traktowane zawsze jako fałsz (bez względu na wartość zmiennej n) i nic nie zostanie wypisane. Instrukcja przypisania `n=0` (użyta zamiast porównania `n==0`) ma sama w sobie wartość taką, jak jej prawa strona (tutaj: 0, co jest równoważne wartości logicznej `false`).

Jeśli kompilator nie wykaże żadnych błędów, to nadal nie możemy mieć pewności, czy nasz program jest zupełnie poprawny. Możemy przecież omyłkowo podzielić coś przez zero, próbować wyciągnąć pierwiastek kwadratowy z liczby ujemnej lub odwołać się do elementu struktury (np. stringu), którego nie ma. Wtedy pojawia się z reguły błąd wykonania (ang. **runtime error**, wymawiaj: *rantajm error*) kwitowany niezbyt miłym komunikatem, który pojawia się w oknie wykonania programu.

*Brak wiadomości to dobra wiadomość.

Niestety nasz kod może zawierać jeszcze inne rodzaje błędów, których przyczyną jest zastosowanie niepoprawnego algorytmu do rozwiązania problemu. No trudno, mówi się, że żywot programisty jest smutny, bo upływa w znacznej mierze na szukaniu błędów w swoich, lub – niestety! – cudzych programach.

Eee, może nie jest aż tak źle...