Pascal

```
program Przyklad8;
var
st, rad: real;
begin
   st := 0;
   repeat
      rad := st * PI / 180;
      Writeln('cos(',st:6:2,') = ', cos(rad):6:2);
      st := st + 10;
   until cos(rad) = 0;
end.
```

C++

```
#include <iostream>
#include <cmath>
using namespace std;
float st, rad;
int main()
{
      st=0.0;
      do
      {
            rad=st*M_PI/180;
            cout << "cos(" << st << ") = " << cos(rad) << endl;
            st+=10.0;
      }
      while(cos(rad)!=0.0);
      return 0;
}
```

Ćwiczenie 6.

W plikach *T7_p8.pas* i *T7_p8.cpp* (CD) zapisane są programy z przykładu 8. Zmodyfikuj wybrany program tak, aby działał poprawnie. Zastanów się, jak najlepiej sformułować warunek w pętli **repeat** (lub **do..while**).

W dalszej części podręcznika omawiany jest algorytm znajdowania miejsca zerowego funkcji metodą połowienia przedziału. W przypadku gdy założona przez użytkownika dokładność otrzymanego wyniku przekroczy możliwości arytmetyki komputerowej, algorytm ten nie zakończy się. Podobny problem występuje w algorytmie wyznaczania pierwiastka kwadratowego.

5. Złożoność obliczeniowa algorytmów

Jak powiedzieliśmy, aby algorytm był poprawny, musi być skończony (czyli gwarantować wyznaczenie rozwiązania w skończonej liczbie kroków). Oczywiste jest, że powinno się to odbyć w jak najkrótszym czasie i przy jak najmniejszym wykorzystaniu zasobów komputera, w szczególności pamięci operacyjnej i masowej. Musimy zatem stworzyć kryterium pozwalające rozstrzygnąć, który z algorytmów służących rozwiązaniu tego samego problemu jest najlepszy.

Kryterium tym jest **złożoność obliczeniowa algorytmu**, którą rozważać będziemy w kontekście czasu wykonania programu (złożoność czasowa) i wykorzystania pamięci przez program (złożoność pamięciowa).