Instrukcja WHILE oraz DO…WHILE

Pętla FOR ma jedną podstawową wadę. Już przed wejściem do pętli musimy ściśle określić **liczbę jej powtórzeń** (za wyjątkiem sytuacji gdy stosujemy **break** lub **continue**). Nie daje się łatwo zrealizować przy jej pomocy czynności wykonywanej „aż do skutku”. Wady tej nie mają pętle: **while** („dopóki…, powtarzaj”) i **do…while** („powtarzaj, aż do…”). A oto ich postacie:

**while** (warunek) **do**

blok\_instrukcji *„Dopóki jest spełniony warunek, wykonuj instrukcje”.*

**do**

blok\_instrukcji

**while** (warunek); *„Wykonuj instrukcje, aż spełniony zostanie warunek”.*

Zasadnicza różnica między tymi pętlami tkwi w momencie sprawdzania warunku. Instrukcja **while** sprawdza warunek przed, a instrukcja **do…while** po wykonaniu bloku instrukcji. Z tego wynika, że blok instrukcji w pętli **do…while** musi się wykonać **conajmniej raz** (dopiero później jest sprawdzany warunek), natomiast w **while** – niekoniecznie (może nie wykonać się nawet **ani razu**).

***Przykłady:***

***Przykład 1*** *Przeanalizuj przykłady z podręcznika do C++.*

***Przykład 2*** *Napisz program, który czyta kolejne liczby i liczy ich sumę, aż do wpisania 0. Wykonaj dwie wersje obliczeń – z while i z do…while.*

cout<<"\nProgram sumuje podawane liczby az do podania zera.";

cout<<"\nZ wykorzystaniem do...while.";

int liczba, suma=0;

do

{

cout<<"\n\nPodaj liczbe: ";

cin>>liczba;

suma+=liczba;

} while (liczba!=0);

cout<<"\nSuma= "<<suma;

cout<<"\nProgram sumuje podawane liczby az do podania zera.";

cout<<"\nZ wykorzystaniem while.";

int liczba, suma=0;

cout<<"\n\nPodaj liczbe: ";

cin>>liczba;

while (liczba!=0)

{

suma+=liczba;

cout<<"\n\nPodaj liczbe: ";

cin>>liczba;

}

cout<<"\nSuma= "<<suma;

***Przykład:*** Napisz program wczytujący z klawiatury liczbę rzeczywistą oznaczającą długośc boku kwadratu. Program powinien przyjmować jedynie liczbę dodatnią. Jeśli użytkownik poda liczbę ujemną, to powinien zostać poinformowany, że wymagana jest liczba dodatnia i poproszony o kolejną liczbę. Próbę wczytywania liczby powtarzamy dopóty, dopóki użytkownik nie poda liczby poprawnej (dodatniej). Taki sposób zapewnienia poprawności wczytywanych danych nazywać będziemy **pętlą zaporową.**

cout<<"\nProgram oblicza pole kwadratu o danym boku.";

cout<<"\nPodaj dlugosc boku: ";

float bok;

cin>>bok;

while (bok<=0) //instrukcja while w roli pętli zaporowej

{

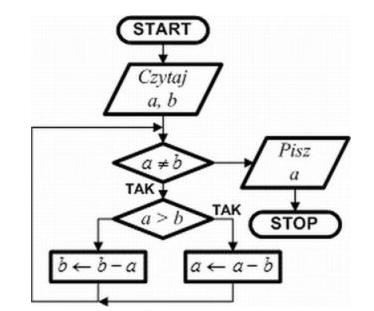
cout<<"\n\nPomyliles sie! Podaj dlugosc boku: ";

cin>>bok;

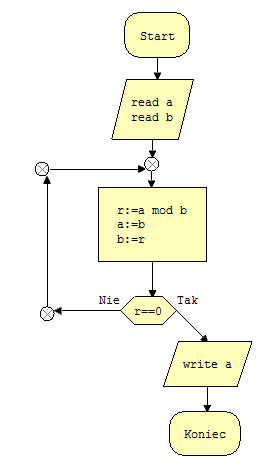
}

cout<<"\n\nBok podany prawidlowo. Pole="<<bok\*bok;

**Zadania**



**Zad.1** ***Algorytm Euklidesa z odejmowaniem.*** Napisz program, który wyznacza NWD dla podanych z klawiatury liczb a i b wg schematu blokowego poniżej. Zastosuj petlę while.



**Zad.2** ***Algorytm Euklidesa z obliczaniem reszty z dzielenia.*** Napisz program, który wyznacza NWD dla podanych z klawiatury liczb a i b wg schematu blokowego poniżej. Zastosuj petlę repeat.

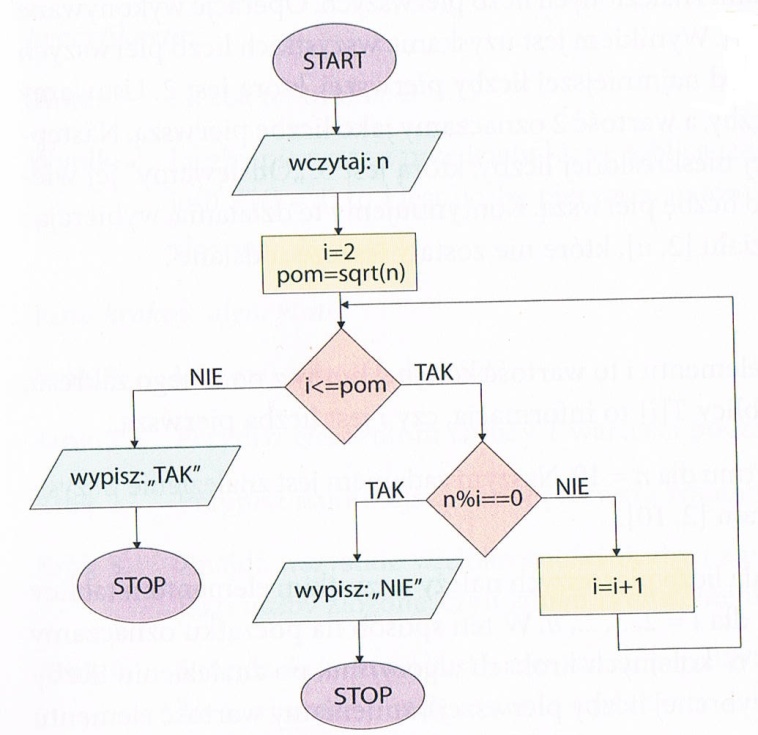
**Zad.3** Dokonaj przeróbki programu z zad.1 lub zad.2 tak, aby obliczał także NWW. Skorzystaj ze wzoru:

**Zad.4** Napisz program, który będzie przeliczał temperaturę podaną w skali Celsjusza na temperaturę w skali Fahrenheita. Program powtarza obliczenia aż do podania -100.

**Zad.5** Napisz program, który będzie wczytywał oceny szkolne ( od 1 do 6) aż do podania 0. Następnie wypisze: ile podano ocen, jaka jest ich średnia arytmetyczna i ile wśród nich było 5-tek. 0 (zero) nie wchodzi do obliczeń!

**Zad.6** Napisz program, który sprawdzi czy podana liczba naturalna n jest liczbą pierwszą.

Liczbę naturalną n większą od 1 nazywamy **liczbą pierwszą**, jeśli posiada **tylko** dwa dzielniki: 1 i n. Liczbę, która nie jest pierwszą nazywamy liczbą złożoną (0 i 1 – nie są ani pierwsze ani złożone).



**Zad.7** Napisz program, który wypisze 100 liczb pierwszych zaczynając od 2 i obliczy ich łączną sumę. Skorzystaj z zad.6.

**Zad.8** Napisz program, który dokona rozkładu liczby naturalnej na czynniki pierwsze. Program powinien podać także ilość różnych czynników w tym rozkładzie.

Przykład: 6 = 2\*3

24 = 2\*2\*2\*3

17 = 17

**Zad.9** Napisz program, który skróci ułamek zwykły (dodatni) oraz wyłączy całości przed ułamek (jeśli to możliwe). Licznik i mianownik ułamka wczytaj z klawiatury.

Np. 9/12 = 3/4 lub 16/6 = 8/3 =2 2/3

**Zad.10** Napisz program, który wczyta dwa ułamki zwykłe, obliczy ich sumę oraz wypisze wynik w postaci ułamka. Jeśli to możliwe skróć ułamek i/lub wyłącz całości.

**Zad.11** Napisz program, który zamieni liczbę arabską na liczbę rzymską. Np. 2014 = MMXIV. Zamień w ten sposób kolejne liczby z przedziału <a;b> wczytanego z klawiatury.