Rozdział 5. Instrukcje języka C++

Instrukcje języka programowania stanowią o tym, co da się w tym języku wyrazić. W tym rozdziale zostaną przedstawione wszystkie instrukcje języka C++ wraz z przykładami. W pierwszej kolejności zostanie omówiona instrukcja podstawienia, bez której nie jest możliwym napisanie żadnego programu. Następnie przedstawione zostaną instrukcje warunkowe (if, switch), w dalszej kolejności instrukcje iteracyjne powszechnie nazywane pętlami (for, while, do...while). Na koniec omówione zostaną rzadko stosowane instrukcje skoku (goto, break, continue).

5.1 Podstawowe instrukcje języka C++

Podstawową instrukcją jest **instrukcja podstawienia:**

Id\_zmiennej=wyrażenie;

Przykładowy kod wykorzystujący powyższą instrukcję przedstawia Listing 5.1

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int i,j,k;
6. k=1;
7. i=(j,j=k++);
8. cout <<"i="<<i<<endl; //i=1
9. cout <<"j="<<j<<endl; //j=1
10. cout <<"k="<<k<<endl; //k=2
11. }

**Listing 5.1**. Kod programu prezentującego zastosowanie instrukcji podstawienia

Mianem **instrukcji złożonej** określamy ciąg instrukcji ujętych w nawiasy syntaktyczne „{” i „}”.

{

Instrukcja1;

Instrukcja2;

…

InstrukcjaN;

}

Jeśli wewnątrz instrukcji złożonej znajduje się deklaracja, to jest to blok. Istotne jest, że jeśli zadeklarujemy jakąkolwiek zmienną w środku bloku to dana zmienna będzie „żyła” tylko w obrębie tego bloku. Przykład przedstawia Listing 5.2.

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int i=2,j=5,k=7;
6. cout <<"i="<<i<<endl; //i=1
7. cout <<"j="<<j<<endl; //j=1
8. {
9. int k=10;
10. int z=1;
11. cout <<"k="<<k<<endl; //k=10
12. cout <<"j="<<j<<endl; //j=5
13. cout <<"z="<<z<<endl; //z=1
14. }
15. cout <<"k="<<k<<endl; //k=7
16. // cout <<"z="<<z<<endl; //błąd!!!!
17. }

**Listing 5.2**. Kod programu prezentującego zakres „życia” zmiennych

Zauważmy, że jeśli „odkomentujemy” linię 16 kodu to kompilator wyrzuci błąd „ *[Error] 'z' was not declared in this scope*”, gdyż zmienna z nie istnieje poza blokiem pomiędzy liniami 8-14.

5. 2 Instrukcje warunkowe

Instrukcje warunkowe są niezbędnym elementem każdego programu, umożliwiają one programiście określenie sposobu reagowania przez program na określone zdarzenia. Jak sama nazwa wskazuje są to instrukcje, które wykonują się tylko wtedy gdy spełniony jest podany warunek. Dzięki instrukcjom warunkowych program może "zachowywać" się zależnie od spełnienia pewnych warunków.

**5.2.1 Instrukcja if**

Składnia instrukcji jest następująca:

if (warunek)

instrukcja1; //instrukcja wykonywana, gdy warunek jest spełniony

else

instrukcja2; //instrukcja wykonywana, gdy warunek nie jest spełniony

**Schemat blokowy** instrukcji if

warunek

instrukcja1

Instrukcja2

TAK

NIE

Instrukcja1 i instrukcja2 mogą być instrukcją złożoną lub blokiem instrukcji, a także inną instrukcją. W takiej sytuacji blok instrukcji składających się na instrukcja1 lub instrukcja2 należy ująć w klamry.

***Zadanie 5.1*** *Napisz program, który po wczytaniu zmiennej całkowitej wypisze resztę z dzielenia przez dziesięć, jeśli wczytana liczba jest większa od 10 i wartość dziesięciokrotnie większą w przeciwnym przypadku.*

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int i;
6. cout<<"i="; cin>>i;
7. if (i>10)
8. {
9. int j;
10. j=i%10;
11. cout<<"j="<<j;
12. }
13. else
14. {
15. int j;
16. j=i\*10;
17. cout<<"j="<<j;
18. }
20. }

**Listing 5.3**. Kod programu realizującego Zadanie 5.1.

Z kolei jeśli Instrukcja1 lub instrukcja2 składają się tylko z jednej linii kodu nawiasy nie są konieczne. Na przykład zbędne są nawiasy klamrowe w sytuacji:

if (x>y)

max=x;

else

max=y;

Z nawiasami i ich brakiem należy jednak uważać spójrzmy na Listing 5.4 Jaki będzie wynik? Dlaczego?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int x=2,y=5,z=7, s=0;
6. if (x>y)
7. s=x+z;
8. else
9. s=x+y;
10. s=y+z;
11. cout<<s;
12. }

**Listing 5.4**. Kod programu prezentującego zastosowanie instrukcji if.

Istnieje również alternatywny zapis umożliwiający wykonanie różnych instrukcji w zależności od spełnienia, bądź nie danego warunku. Zapis ten wykorzystuje operator warunkowy i ma postać:

( <warunek> ) ? <wyrażenie1> : <wyrażenie2>;

Jeżeli spełniony jest <warunek> to operator zwróci wartość <wyrażenia1>, w przeciwnym wypadku wartość <wyrażenia2>.

UWAGA: <wyrażenie1> nie jest zakończone średnikiem natomiast <wyrażenie2> może (a nawet musi) być zakończone średnikiem gdy wartość zwracana przez operator nie jest przekazywana jako argument.

***Zadanie 5.2*** *Napisz program wypisujący czy wczytana liczba całkowita jest, czy nie jest parzysta****.***

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int x;
6. cout << "Podaj liczbę całkowitą:" << endl;
7. cin >> x;
8. cout << ( (x & 1) ? "Liczba nieparzysta." : "Liczba parzysta.") << endl;
9. }

**Listing 5.5**. Kod programu realizującego Zadanie 5.2.

Oczywiście możemy stosować również kilka warunków do sprawdzenia, gdy pierwszy nie jest spełniony. Instrukcja if z wielokrotnym sprawdzeniem warunków ma postać:

if (warunek1)

instrukcja1; //instrukcja wykonywana, gdy warunek1 jest spełniony

else if (warunek2)

instrukcja2; //instrukcja wykonywana, gdy warunek2 jest spełniony

else

instrukcja3; //instrukcja wykonywana, gdy warunek1 i warunek2 nie są spełnione

**Schemat blokowy**

Warunek1

Instrukcja3

TAK

NIE

Warunek2

NIE

TAK

Instrukcja2

W przypadku, kiedy mamy do sprawdzenia wiele warunków alternatywnych złożona instrukcja if może być bardzo rozbudowana. W takiej sytuacji, często bardziej czytelnym jest zastosowanie instrukcji switch.

**5.2.2 Instrukcja switch**

W sytuacji, gdy mamy dużo warunków nałożonych na jedną zmienną typu całkowitego można zastosować instrukcje switch. Sprawdza ona jaką wartość ma zmienna, a następnie wykonuje przypisane dla tej wartości kolejne instrukcje. Jej składnia jest następująca:

switch (wyrażenie całkowite)

{

case w1: instrukcja1;

instrukcja2;

...

instrukcjaN;

break;

case w2: instrukcja1;

instrukcja2;

...

instrukcjaN;

break;

...

default: instrukcja1;

instrukcja2;

...

instrukcjaN;

}

Działanie instrukcji switch jest następujące:

* Obliczamy wartość wyrażenia całkowitego,
* Jeśli jest równa w1, to wykonywane są instrukcje z grupy w1,
* Jeśli jest równa w2, to z grupy w2, itd.
* Jeśli wartość wyrażenia całkowitego nie przyjmuje wartości ani w1, ani w2 itd., to są wykonywane instrukcje po słowie default.

Sytuacje wyjątkowe:

* Instrukcja break nie musi wystąpić, wtedy są wykonywane instrukcje do napotkania następnego słowa break lub do końca,
* Słowo kluczowe default nie musi wystąpić. Jeśli go nie ma i wartość wyrażenia całkowitego nie przyjmuje jednej z wartości w1, w2, …, to instrukcja switch nie zmienia stanu komputera,
* Jeśli dla pewnego zbioru wartości chcemy wykonać żądane operacje, to używamy zapisu:
  + case w1: case w2: case w3: instrukcje;

***Zadanie 5.3*** *Napisz program informujący czy dana liczba jest podzielna przez 5 lub jaka jest reszta z dzielenia jej przez 5. Program napisz w dwóch wersjach: z wykorzystaniem instrukcji if lub switch.*

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int i;
6. cout<<"i="; cin>>i;
7. if ((i%5==1)||(i%5==3))
8. cout<<"reszta z dzielenia wynosi 1 lub 3";
9. else if (i%5==2)
10. cout<<"reszta z dzielenia wynosi 2";
11. else if (i%5==4)
12. cout<<"reszta z dzielenia wynosi 4";
13. else
14. cout<<"liczba podzielna przez 5";
15. }

**Listing 5.6**  Kod programu realizującego Zadanie 5.3 z wykorzystaniem instrukcji if.

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int i;
6. cout<<"i="; cin>>i;
7. switch (i%5)
8. {
9. case 1: case 3:
10. cout<<"reszta z dzielenia wynosi 1 lub 3";
11. break;
12. case 2:
13. cout<<"reszta z dzielenia wynosi 2";
14. break;
15. case 4:
16. cout<<"reszta z dzielenia wynosi 4";
17. break;
18. default:
19. cout<<"liczba podzielna przez 5";
20. }
21. }

**Listing 5.7**  Kod programu realizującego Zadanie 5.3 z wykorzystaniem instrukcji switch.

5.3 Instrukcje iteracyjne (pętle)

Instrukcje iteracyjne służą do wykonywania pewnych sekwencji instrukcji programu określoną ilość razy.

**5.3.1 Instrukcja pętli for**

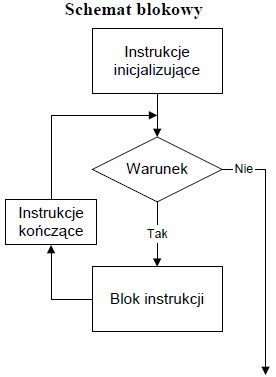
Pętli for używamy wtedy, chcemy wykonać instrukcje ustaloną liczbę razy.

Składnia:

for (sekcja\_inicjalizacji; sekcja\_warunku;  
 sekcja\_modyfikacji\_zmiennej\_sterującej)

Instrukcja;

gdzie Instrukcja może być instrukcją prostą, złożoną lub blokiem.



Działanie instrukcji for:

1. wykonywana jest inicjalizacja,
2. sprawdzany jest warunek,
3. jeśli warunek jest prawdziwy, to wykonywana jest instrukcja,
4. wykonywana jest modyfikacja zmiennej sterującej,
5. następuje powrót do punktu 2.

Sytuacje wyjątkowe:

* dowolna sekcja może być pusta, wtedy ich zadanie muszą przejąć albo instrukcje przed pętlą (dla inicjalizacji) lub ciało instrukcji pętli,
* można kontrolować wiele zmiennych sterujących, np. inicjowanie dwu zmiennych sterujący jest możliwe następująco: int i=0, j=10,
* można w warunku zapisywać kontrolę wielu zmiennych, używamy wtedy wyrażeń logicznych, np.: (i<=10) && (j>=0)
* można modyfikować wiele zmiennych sterujących, np. i++, j—

***Zadanie 5.4*** *Napisz program, który wypisze stablicuje wartości funkcji y=exp(x) z krokiem 0.5 w przedziale [1, n], gdzie n jest podane przez użytkownika. W przypadku, gdy wartość bezwzględna argumentu funkcji przekroczy 10 program wypisze informację, że wartość jest zbyt duża.*

1. #include <iostream>
2. #include <math.h>
3. using namespace std;
4. int main()
5. {
6. int n;
7. cout<<"n="; cin>>n;
8. for (float x=1; x<=n; x=x+0.5)
9. {
10. if (fabs(x)<=10)
11. {
12. float y=exp(x);
13. cout <<x<<"\t"<<y<<endl;
14. }
15. else
16. cout<<"za duza wartość";
17. }
18. }

**Listing 5.8**  Kod programu realizującego Zadanie 5.4

**5.3.3 Instrukcja pętli while**

Kolejnym przykładem pętli jest pętla while, w przeciwieństwie do pętli for w niej nie określamy ilości powtórzeń, natomiast będzie się ona wykonywała tak długo, aż wskazany warunek nie zostanie spełniony - tak więc w przeciwieństwie do pętli for nie jest sterowana licznikiem, tylko warunkiem, lecz podobnie jak tamta może się nie wykonać ani razu.

Składnia:

while(warunek){

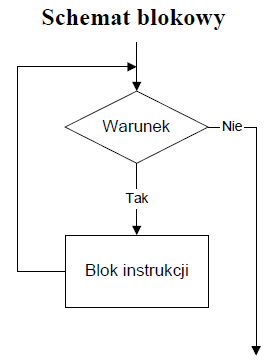
instrukcja1;

instrukcja2;

...

instrukcjaN;

}



Działanie instrukcji while:

* Sprawdzany jest warunek.
* Jeśli warunek jest fałszywy blok instrukcji nie jest wykonywany i działanie pętli jest zakończone.
* Jeśli warunek jest prawdziwy wykonywane są instrukcje, a następnie ponownie następuje przejście do punktu 1 i sprawdzany jest warunek.

Ogólnie można powiedzieć, że blok instrukcji jest wykonywany tak długo jak warunek jest spełniony.

***Zadanie 5.5.*** *Napisz program wczytujący z klawiatury liczbę całkowitą. Program powinien przyjmować jedynie liczbę dodatnią, w przypadku podania liczby ujemnej program informuje użytkownika, że wymagana jest liczba dodatnia i prosi o kolejną liczbę. Próbę wczytywania liczby powtarzamy dopóty, dopóki użytkownik nie poda liczby poprawnej (dodatniej). Następnie program powinien zwrócić sumę cyfr podanej liczby.*

1. #include <iostream>
2. #include<math.h>
3. using namespace std;
4. int main()
5. {
6. int n;
7. cout << "Podaj liczbę dodatnia";
8. cin >> n;
9. while (n<=0)
10. {
11. cout << "Podałeś liczbę ujemną!\n\nPodaj liczbę dodatnią ";
12. cin >> n;
13. }
14. int absn=abs(n); // wartość bezwzględna z n
15. int suma=0;
16. while(absn > 0) //dopóki są jeszcze cyfry
17. {
18. suma += absn % 10; //do sumy dodajemy ostatnią cyfrę liczby absn
19. absn /= 10; //liczbę absn dzielimy przez 10, skracamy w ten sposób   
     ostatnia cyfrę
20. }
21. cout << "Suma cyfr liczby " << n << "wynosi " << suma << endl;
22. system("PAUSE");
23. }

**Listing 5.9**  Kod programu realizującego Zadanie 5.5 w wykorzystaniem instrukcji while.

**5.3.3 Instrukcja pętli do...while**

Pętla do..while różni się od pętli while tym, że musi zostać wywołana przynajmniej jeden raz. Spowodowane to jest tym, że najpierw wykonywany jest blok instrukcji, a dopiero później sprawdzany zostaje warunek. W odróżnieniu od poprzedniczek jest to pętla, którą należy zakończyć średnikiem. Jest to spowodowane tym, że kończy się komendą, a nie znacznikiem zamknięcia bloku.

Składnia:

do

{

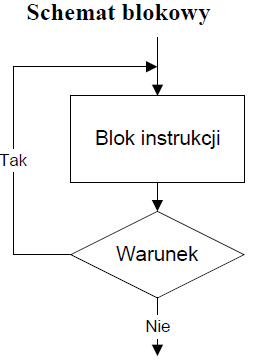
instrukcja1;

instrukcja2;

…

instrukcjaN;

} while (warunek);



Działanie pętli do … while:

1. Wykonywany jest blok instrukcji,
2. Sprawdzany jest warunek,
3. jeśli warunek jest prawdziwy, to powtarzamy od punktu 1, w przeciwnym razie pętla jest kończona.

Uwaga: Jeśli wykonywana jest pojedyncza instrukcja, to nawiasy klamrowe można (i należy) opuścić.

W większości programów instrukcje while i do...while można stosować wymiennie wprowadzając do kodu pewne zmiany. I tak Zadanie 5.5 można napisać wykorzystując pętlę do..while. Rozwiązanie przedstawia Listing 5.10.

1. #include <iostream>
2. #include<math.h>
3. using namespace std;
4. int main()
5. {
6. int n;
7. do{
8. cout << "Podaj liczbe dodatnia";
9. cin >> n;
10. }while (n<=0);
11. int absn=abs(n); // wartość bezwzględna z n
12. int suma=0;
13. do{
14. suma += absn % 10; //do sumy dodajemy ostatnią cyfrę liczby absn
15. absn /= 10; //liczbę absn dzielimy przez 10, skracamy w ten sposób   
     ostatnia cyfrę
16. }
17. while(absn > 0); //dopóki są jeszcze cyfry
18. cout << "Suma cyfr liczby " << n << "wynosi" << suma << endl;
19. system("PAUSE");
20. }

**Listing 5.10**  Kod programu realizującego Zadanie 5.5 w wykorzystaniem instrukcji do...while.

Porównując rozwiązania z Listingu 5.9 i 5.10 należy zwrócić uwagę na to, że w przypadku zastosowania pętli while zmienna n musi być wczytana przed wejściem do pętli, natomiast w przypadku pętli do..while zmienna n może być wczytana w pętli, gdyż ta pętla zawsze wykona się co najmniej raz.

***Zadanie 5.6*** *Napisz program wykonujący podstawowe działania arytmetyczne dwóch liczb tj. Dodawanie i odejmowanie. Umożliw użytkownikowi wielokrotny wybór działania.*

1. #include <iostream>
2. #include <math.h>
3. using namespace std;
4. int main()
5. {
6. int n,l1,l2;
7. do
8. {
9. cout<<"1.Dodawanie"<<endl;
10. cout<<"2.Odejmowanie"<<endl;
11. cout<<"3.Koniec"<<endl;
12. cin>>n;
13. switch(n)
14. {
15. case 1:
16. cout<<"l1="; cin>>l1;
17. cout<<"l2="; cin>>l2;
18. cout<<"l1+l2="<<l1+l2<<endl;
19. break;
20. case 2:
21. cout<<"l1="; cin>>l1;
22. cout<<"l2="; cin>>l2;
23. cout<<"l1-l2="<<l1-l2<<endl;
24. break;
25. }
26. } while(n!=3);
27. }

**Listing 5.11**  Kod programu realizującego Zadanie 5.6

5.4. Instrukcje skoku

* + 1. **Instrukcja goto**

Skok bezwarunkowy **goto** jest najstarszym rodzajem pętli, który został wyparty przez pętlę **for** i pętlę **while.** Do konstrukcji pętli **goto** potrzebna jest etykieta, czyli takie słowo, które jest zakończone znakiem dwukropka (:). Etykieta musi występować w linii jako pierwsza. Program gdy napotka słowo kluczowe **goto** i nazwę tejże etykiety, przemieszcza się bezwarunkowo w miejsce gdzie  została ona zadeklarowana.

Nie jest wskazane stosowanie tego rodzaju pętli, ponieważ kod staje się nieczytelny, trudny do analizy oraz łamie ideę programowania strukturalnego.

Używanie instrukcji skoku nie jest zbyt skomplikowane. Po pierwsze, aby móc gdzieś skoczyć musimy wyznaczyć miejsce do tego, deklaruje się tzw. etykietę, możemy ją dowolnie nazwać   
(np: etykieta\_skoku) i umieścić ją w dowolnym miejscu danej funkcji(bloku).

...  
etykieta\_skoku:  
...

Po prostu pisze się jej nazwę, a potem dwukropek. Aby skoczyć do danej etykiety, trzeba napisać instrukcję goto (idź do) i po spacji podać nazwę etykiety, do której mamy skoczyć. Należy zakończyć taką instrukcję średnikiem:

...  
**goto** etykieta\_skoku;  
...

WAŻNE: instrukcja skoku może działać, tylko w tym samym bloku (funkcji), nie możemy przeskoczyć do innej funkcji!

Przykład programu wykorzystującego instrukcję goto przedstawia Listing 5.12

1. #include <iostream>
2. #include <math.h>
3. using namespace std;
4. int main()
5. {  
     char litera;
6. cout<<"Czy chcesz przeskoczyc dalej?\nPodaj t dla tak\n";
7. cin>>litera;
8. if(litera=='t') goto etykieta\_skoku;
9. cout<<"Wykonuja sie instrukcje ktorych nie przeskoczyles\n";
10. etykieta\_skoku:
11. system("PAUSE");
12. }

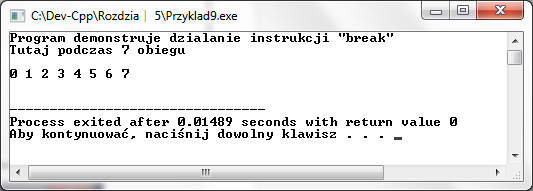
**Listing 5.12** Kod przedstawiający działanie instrukcji goto.

* + 1. **Instrukcja break**

Instrukcja break pozwala na **przerwanie działanie jednej z pętli: for, while, do...while oraz instrukcji switch.** Powoduje wyjście z najbardziej zagnieżdżonej pętli lub przerwanie wykonywania instrukcji [switch](http://eff10.internetdsl.tpnet.pl/programowanie/brzozowski/cpp/instrukcje.html#i). Przykład programu wykorzystującego instrukcję break przedstawia Listing 5.13

1. #include<iostream>
2. using namespace std;
3. int main()
4. {
5. unsigned i;
6. cout<<"Program demonstruje dzialanie instrukcji \"break\""<<endl;
7. cout<<"Tutaj podczas 7 obiegu"<<endl<<endl;
8. for(i=0; i<100; i++)
9. {
10. cout<<i<<" ";
11. // Po wypisaniu siedmiu liczb kończymy...
12. if( i>=7 )
13. {
14. cout<<endl<<endl;
15. break;
16. }
17. }
18. }

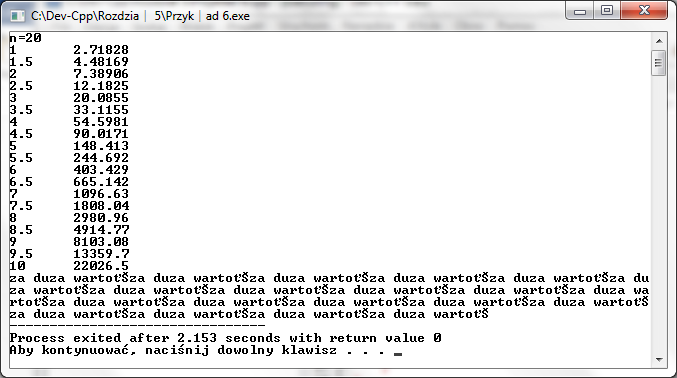
**Listing 5.13** Kod przedstawiający działanie instrukcji break.



**Rysunek 5.1**. Wynik działania programu przedstawionego na Listingu 5.13.

***Zadanie 5.7.*** *Zoptymalizuj program z Listingu 5.8 wykorzystując instrukcję break.*

Zastosowanie funkcji break umożliwi nam bardziej efektywne napisanie programu rozwiązującego Zadanie 5.4. Zauważmy bowiem, że przy podaniu dużej liczby n kod przedstawiony na Listingu 5.8 wielokrotnie wypisuje informację "za duza wartość" zakodowaną w linii 16 kodu. Przykładowe uruchomienie przedstawia Rysunek 5.2.



**Rysunek 5.2**. Wynik działania programu przedstawionego na Listingu 5.8.

Wystarczyłoby aby informacja o tym, że mamy „za dużą wartość” pojawiła się jednokrotnie i pętla for nie byłaby wykonywana dalej. Możemy to poprawić wykorzystując instrukcje break w sytuacji, gdy fabs(x)>10. Zoptymalizowany kod przedstawia Listing 5.14.

1. #include <iostream>
2. #include <math.h>
3. using namespace std;
4. int main()
5. {
6. int n;
7. cout<<"n="; cin>>n;
8. for (float x=1; x<=n; x=x+0.5)
9. {
10. if (fabs(x)<=10)
11. {
12. float y=exp(x);
13. cout <<x<<"\t"<<y<<endl;
14. }
15. else
16. {
17. cout<<"za duza wartość";
18. break;
19. }
20. }
21. }

**Listing 5.14.** Kod programu realizującego Zadanie 5.6

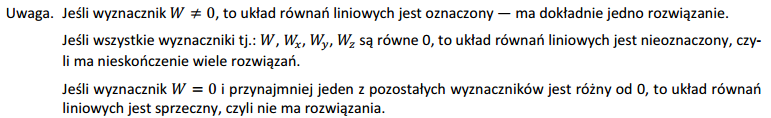
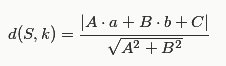
* + 1. **Instrukcja continue**

Instrukcja continue, podobnie jak break, używana jest tylko wewnątrz pętli do-while, while i for. Pomija ona resztę ciała pętli, powodując natychmiastowe rozpoczęcie kolejnego cyklu pętli. To odróżnia ją od instrukcji break, która całkowicie wyskakuje z pętli. Przykład programu wykorzystującego instrukcję continue przedstawia Listing 5.15

1. #include<iostream>
2. using namespace std;
3. int main()
4. {
5. unsigned i;
6. cout<<"Program demonstruje dzialanie instrukcji \"continue\""<<endl;
7. cout<<"Dla i nieparzystych ( i%2 = 1 ) pominiemy wypisanie liczby"<<endl<<endl;
8. for(i=0; i<10; i++)
9. if( i%2 )
10. continue;
11. else
12. cout<<i<<" ";
13. cout<<endl;
14. }

**Listing 5.15** Kod przedstawiający działanie instrukcji continue.

5.4 Zadania

1. Dla podanej liczby n (n z przedziału (0,1000>) wyznaczyć i wyświetlić n kolejnych wyrazów ciągu arytmetycznego o zadanych przez użytkownika pierwszym elemencie a i różnicy r, gdzie a jest liczbą całkowitą z przedziału <0,20>, r jest liczbą całkowitą różną od 0 z przedziału  
    <-10000,10000>. Obliczyć sumę tych elementów.
2. Napisz program, który w zależności od wybranej opcji dokonuje zamiany podanej liczby naturalnej n podanej w systemie dziesiętnym, na liczbę w systemie binarnym lub liczby w systemie binarnym b, na liczbę w systemie dziesiętnym. Liczba n podawana jest przez użytkownika i mieści się w przedziale <0,255>, natomiast liczba b jest ciągiem zer i jedynek nie dłuższym niż 8. Jeśli użytkownik poda nieprawidłowe dane powinien zostać wyświetlony odpowiedni komunikat i wczytanie podanie nowych danych.
3. Napisz program, w którym wczytasz liczbę naturalną n, a następnie wczytując kolejno n liczb naturalnych obliczysz ile wśród wczytanych liczb jest takich, które: są podzielne przez 3 i 2,  
    a niepodzielne przez 5.
4. Napisz program, w którym wczytasz liczbę naturalną n, a następnie wczytując kolejno n liczb naturalnych obliczysz ile wśród wczytanych liczb jest takich, które spełniają warunek |ak| < k2, , gdzie k oznacza numer liczby w kolejności.
5. Napisz program, który wczyta liczbę X oraz wyliczy i wyświetli kolejne elementy ciągu Fibonacciego, które nie przekraczają wartości wczytanej zmiennej X. W przypadku, gdy kolejny wyliczony element jest większy od zmiennej X, program powinien zakończyć działanie. Wykorzystaj pętlę while.
6. Dla wczytanej liczby n wypisz dla tabliczkę mnożenia lub dzielenia (całkowitego lub rzeczywistego).
7. Napisz program, który wyznaczy rozwiązania układu równań z trzema niewiadomymi metodą wyznaczników. Sprawdzi czy jest jedno rozwiązanie, nieskończenie wiele rozwiązań czy też układ jest sprzeczny. 
8. Dany jest ciąg liczb naturalnych. Napisz program, który wypisze te liczby w zapisie rzymskim.   
   np.: 50, 25, 600, 14, 90, -5  
   L, XXV, DC, XIV, XC
9. Napisz program symulujący działanie kalkulatora. Kalkulator ma pracować w dwu trybach: standardowym i profesjonalnym. W trybie standardowym kalkulator ma operować na liczbach rzeczywistych i uwzględniać operacje: dodawanie, odejmowanie, mnożenie i dzielenie a w trybie rozszerzonym ma ponadto obliczać funkcje matematyczne: sin, cos, exp i losować liczbę rzeczywistą pseudolosową z zakresu 0...1.
10. Zmodyfikuj działanie kalkulatora z zadania 1 w taki sposób, by można było wielokrotnie wykonywać obliczenia. Przerwanie pracy programu po wprowadzeniu 0.
11. Wczytaj trzy liczby rzeczywiste oznaczające długości boków trójkąta tak by mogły one utworzyć trójkąt na płaszczyźnie. Użyj pętli do .. while.
12. Wylosuj cyfrę z zakresu 0..9 – ma być ona ukryta przed użytkownikiem programu. Spróbuj odgadnąć tą liczbę w trzech próbach. Zakończenie programu powinno wyświetlić komunikat: „PIN poprawny” lub „PIN niepoprawny”. Użyj pętli do … while.
13. Dany jest ciąg n-elementowy. Napisz program obliczający wariancję i odchylenie standardowe korzystając z trzech różnych pętli: (1) for, (2) while i (3) do … while.   
    W obliczeniach wykorzystaj wzory:  
    - wariancja; - odchylenie standardowe.  
    Jakie są wady i zalety poszczególnych rozwiązań?
14. Napisz program, w którym wczytasz liczbę naturalną n, a następnie wczytując kolejno n liczb naturalnych obliczysz ile wśród wczytanych liczb jest takich, które są kwadratami liczby parzystej.
15. Napisz program, który wczyta N liczb całkowitych i wyznaczy iloczyn tych liczb, które są parzyste oraz sumę liczb nieparzystych.
16. Dane są dwie liczby całkowite dodatnie. Napisz program, który obliczy największy wspólny dzielnik podanych liczb, wykorzystaj algorytm Euklidesa. Zwróć uwagę na wydajność algorytmu dla danych np. 1 i 10000.
17. Należy napisać program, który w zależności od wybranej opcji dokonuje zamiany podanej liczby naturalnej n podanej w systemie dziesiętnym, na liczbę w systemie szesnastkowym lub liczby w systemie szesnastkowym s, na liczbę w systemie dziesiętnym. Liczba n podawana jest przez użytkownika i mieści się w przedziale <0,65535>, natomiast liczba s jest ciągiem cyfr 0-9 lub liter   
    A-F nie dłuższym niż 4. Jeśli użytkownik poda nieprawidłowe dane powinien zostać wyświetlony odpowiedni komunikat i wczytanie podanie nowych danych.
18. Napisz program Rok przestępny. Program pobiera z klawiatury numer roku i drukuje na ekranie informacje ROK JEST PRZESTĘPNY lub ROK NIE JEST PRZESTĘPNY
19. Napisz program wczytujący z klawiatury liczbę naturalną i wyświetlający na ekranie komunikat określający, czy wprowadzona liczba jest liczbą pierwszą. Wskazówka: liczba pierwsza to liczba naturalna większa od 1, podzielna tylko przez 1 i przez samą siebie, np. 2, 3, 5, 7, 11, 13, 17, 19, 23, 27 itd.
20. Napisz program wyświetlający na ekranie wartości funkcji f(x)=1/x w zadanym przedziale [a, b] z krokiem k (tzw. tablicowanie funkcji).
21. Napisz program, który w zależności od wyboru użytkownika wczytuje wymiary: kwadratu, prostokąta lub trójkąta i wypisuje na ekran pole figury o wczytanych wymiarach.
22. Napisz program, w którym wczytasz liczbę naturalną n, a następnie wczytując kolejno n liczb rzeczywistych znaleźć ilość sąsiadujących par (a; b) takich, że a > 0 i b > 0. (Przykład: dla liczby n = 6 i kolejnych liczb 3; 5; 2;-4; 9; 7 poprawna odpowiedz to 3 (pary (3; 5), (5; 2) oraz (9; 7))).
23. Napisz program, w którym wczytasz liczbę naturalną n, a następnie wczytując kolejno n liczb naturalnych obliczysz ile wśród wczytanych liczb jest takich, które spełniają warunek   
    2k < ak < k!, gdzie k oznacza numer liczby w kolejności.
24. Napisz program, w którym wczytasz liczbę naturalną n, a następnie wczytując kolejno n liczb naturalnych obliczysz ile wśród wczytanych liczb jest takich, które spełniają warunek   
    ak < (ak-1+ak+1)/2, gdzie k oznacza numer liczby w kolejności.
25. Napisz program, który wczytuje liczbę n i wypisuje na ekran wszystkie trojki pitagorejskie   
    (tj. trojki liczb całkowitych a, b, c takich, że a2 +b2 = c2), składające się z liczb mniejszych od n.
26. Napisz program generujący n liczb pseudolosowych. Pierwszą liczbą powinna być dowolna losowa liczba z przedziału (0, 1). Kolejne liczby powinny być wyliczane ze wzoru xn = 1 − x2 n−1,  
    gdzie xn to bieżąca, a xn−1 to poprzednia wartość liczby.
27. Napisz program, który z wykorzystaniem instrukcji switch oblicza pierwiastki równania kwadratowego ax2+bx+c = 0, w którym zmienne a, b, c to liczby rzeczywiste wprowadzane z klawiatury.
28. Napisz program, który generuje losową liczbę naturalną, która ma zostać odgadnięta. Następnie w pętli do – while użytkownik próbuje odgadnąć liczbę podając swoja propozycję.   
    Jeśli zaproponowana liczba jest odpowiednio mniejsza lub większa –wyświetlany jest odpowiedni komunikat. Pętla kończy się w momencie trafienia w szukaną liczbę.
29. Napisz program, który z ciągu n-liczb całkowitych wprowadzonych z klawiatury wypisze te, których najbardziej i najmniej znacząca cyfra są takie same.  
    np. -131, 25672
30. Napisz program, który z ciągu n-liczb całkowitych wprowadzonych z klawiatury wypisze te, które spełniają następujący warunek: naprzemiennie występują cyfry parzyste i nieparzyste  
    Np. 121438, -8783
31. Dany jest ciąg liczb naturalnych wprowadzonych z klawiatury, zakończony zerem. Napisz program, który wypisze kolejne liczby ciągu w postaci iloczynu liczb pierwszych.  
    Np. Dla ciągu liczb: 24, 100, 6, 1520, 0  
    2\*2\*2\*3  
    2\*2\*5\*5  
    2\*3  
    2\*2\*2\*2\*5\*19
32. Dany jest ciąg liczb całkowitych zakończony zerem. Napisz program, który wypisze liczby których cyfry tworzą ciąg niemalejący, od najbardziej znaczącej do najmniej znaczącej.  
    np. 14456, 7881, -13, 8889, 252  
    wynik: 14456, -13, 8889
33. Napisz program, który z ciągu n-liczb całkowitych wprowadzonych z klawiatury wypisze te, w których nie powtarzają się cyfry.
34. Napisz program zgadywanie. Użytkownik podaje początkowy przedział z jakiego wymyślił liczbę a zgadującym jest komputer. Zgaduje losując liczby z przedziału wyznaczonego na podstawie poprzednich odpowiedzi (np. komputer początkowo losował z przedziału[0..20] podał liczbę 10 i dostał odpowiedź od użytkownika: *Za mało*, to następne losowanie komputera będzie z przedziału [10..20]). Gra toczy się do momentu odgadnięcia liczby przez komputer.
35. Napisz program, który wczyta równanie prostej Ax+By+C=0 oraz okręgu  
     (x- a )2 +(y-b)2=r2 a następnie zbada ich wzajemne położenie.  
    Uwaga ! Prosta i okrąg: mają dwa punkty wspólne, jeżeli d(S,k)<r; są styczne, jeżeli d(S,k)=r ; nie przecinają się, jeżeli d(S,k)>r. d(S,k)-oznacza odległość środka okręgu od prostej k : 
36. Napisz program, który wczytuje długości boków trójkąta i sprawdza, czy trójkąt ten jest równoboczny, równoramienny, czy też różnoboczny.
37. Napisz program, który sprawdza kąt podany przez użytkownika (w stopniach, między 0 a 360 stopni), czy jest on wypukły, półpełny czy wklęsły; jeżeli jest wypukły, to sprawdza, czy jest ostry, prosty czy rozwarty, jeżeli jest ostry, to sprawdza czy jest zerowy, i wreszcie, jeżeli jest wklęsły, to sprawdza, czy jest pełny.
38. Dany jest ciąg liczb zakończony zerem. Napisz program, który w danym ciągu wyszuka liczby podzielne przez 3 a niepodzielne przez 6. Program powinien wypisać znalezione liczby lub komunikat o braku takich liczb w ciągu.
39. Dane są trzy punkty: (x0, y0), (x1, y1), (x2, y2), gdzie x są argumentami a y są wartościami paraboli o nieznanych współczynnikach. Napisz program, który na ich podstawie obliczy współczynniki a, b, c wielomianu 2-go stopnia, a następnie obliczy 5 wartości tego wielomianu dla wybranych x.
40. Dane są dwa punkty w przestrzeni **R**3 P1 i P2 oraz równanie prostej, która przechodzi przez te dwa punkty postaci: P(t)= tP1 + (1-t)P2, dla 0<=t<=1 jest to równanie odcinka. Napisz program, który wypisze n punktów równoodległych pomiędzy P1 i P2.
41. Dana jest liczba n oznaczająca liczbę cyfr w liczbie, n<=5. Napisz program, który wylosuje cyfry dla tej liczby oraz wypisze jej wartość w systemie dwójkowym.
42. Napisz program, który obliczy pochodną funkcji f(x)=x^n na podstawie jej definicji: f'(x)=lim(f(x+h)-f(x))/h. Sprawdź wpływ typu zmiennej (float, double) h na dokładność obliczeń. Porównaj z dokładną wartością pochodnej: f'(x)=nx^(n-1).
43. Napisz program drukujący na ekranie wieżę prostokąt z literek S. Rozmiar podstawy podaje użytkownik. Np. wieża o podstawie długości 4:  
    S  
    SS  
    SSS  
    SSSS
44. Napisz program drukujący na ekranie wieżę z literek U. Długość obu boków podaje użytkownik. Np. Prostokąt o bokach długości 5 i 4:  
    UUUUU  
    U U  
    U U  
    UUUUU
45. Napisz program drukujący na ekranie choinkę z znaku #. Wysokość choinki podaje użytkownik. Np. Choinka o wysokości 5 :  
     #  
     ###  
     #####  
     #######  
    #########