Temat C1. Więcej o stosowaniu instrukcji iteracyjnych w językach C++ i Python

1. **Stosowanie instrukcji iteracyjnej `for` w językach C++ i Python – powtórzenie**
2. **Stosowanie instrukcji `while` w językach C++ i Python**
3. **Stosowanie instrukcji `do ... while` w języku C++**

Warto powtórzyć

1. Na czym polega iteracja?
2. Jak można zapisać algorytm iteracyjny w wybranym języku programowania (C++ lub Python)?
3. Jak określamy liczbę iteracji w instrukcji `for` w wybranym języku programowania (C++ lub Python)?
4. W jaki sposób określa się krok iteracji w wybranym języku programowania (C++ lub Python)?
5. Jak działa instrukcja warunkowa w wybranym języku programowania (C++ lub Python)?
6. W jaki sposób oznaczamy blok instrukcji w wybranym języku programowania (C++ lub Python)?

1. Stosowanie instrukcji iteracyjnej `for` w językach C++ i Python – powtórzenie

Zbierasz pieniądze na zakup nowego komputera. Wpłacasz co miesiąc dowolną kwotę na swoje konto. Chcesz obliczyć sumę pieniędzy wpłaconych w ciągu roku. Jak zapisać rozwiązanie tego zadania w wybranym języku programowania?

W przedstawionym problemie powtarzają się dwie czynności: wprowadzania wpłacanej kwoty i powiększania sumy pieniędzy o tę kwotę. W językach programowania do zapisania powtarzających się poleceń wykorzystujemy instrukcje iteracyjne (zwane również instrukcjami pętli).

Aby zapisać algorytm iteracyjny w językach C++ i Python, stosujemy instrukcję iteracyjną **`for`**. W tworzonych przez nas programach liczba powtórzeń (iteracji) będzie z góry określona lub wprowadzana z klawiatury po uruchomieniu programu.

Instrukcja iteracyjna **for**	**for** *(wyrażenie_początkowe; warunek; wyrażenie_pętli)* *lista_instrukcji;* *kolejna_instrukcja;*	C++
	for *zmienna* **in** *lista_wartości:* *lista_instrukcji* *kolejna_instrukcja*	Python

Tabela 1. *Ogólna postać instrukcji iteracyjnej* **for** *w językach C++ i Python*

Lp.	C++	Python
1.	`for (i = 0; i < 50; i++)` `  cout << i << endl;`	`for i in range(50):` `  print(i)`
2.	`for (i = 0; i <= n; i++)` `  cout << "#" << endl;`	`for i in range(n+1):` `  print("#")`

3.	`for (i = 1; i < k; i++)`
	`{`
	`  cout << "Podaj liczbe: ";`
	`  cin >> a;`
	`  cout << a * a;`
	`}`
		`for i in range(1, k):`
		`  a = int(input("Podaj liczbę: "))`
		`  print(a * a)`

4.	`for (i = 100; i > 0; i--)` `  cout << i << endl;`	`for i in range(100, 0, -1):` `  print(i)`

Tabela 2. *Przykłady stosowania instrukcji iteracyjnej* **for** *w językach C++ i Python*

Ćwiczenie 1. Analizujemy działanie instrukcji iteracyjnej `for`

1. Przeanalizuj zapisy podane w przykładzie 1., a następnie odpowiedz na pytania:
   a. Jakie wartości przyjmuje zmienna `i`? C+ i Python
   b. Ile razy zostanie wykonana *lista_instrukcji*? C+ i Python
   c. Co oznacza zapis `i++` i `i--`? C++
   d. Co określają parametry funkcji `range()`? Python

Ćwiczenie 2. Stosujemy instrukcję iteracyjną `for`

1. Napisz specyfikację zadania i program, który wyświetli w kolumnie liczby naturalne od 1 do *n*, i dalej, w tej samej kolumnie – liczby od *n* do 1.
2. Zapisz program w pliku pod nazwą *Liczby_rosnaco_malejaco*.

Uwaga

W języku C++ należy pamiętać o skompilowaniu programu przed jego uruchomieniem – program nie zostanie uruchomiony, jeśli kompilacja nie przebiegła prawidłowo.

Ćwiczenie 3. Obliczamy sumę dwunastu liczb wprowadzanych z klawiatury

Napisz specyfikację zadania i program, który obliczy sumę dwunastu comiesięcznych wpłat na zakup komputera i wyświetli wynik na ekranie. Zapisz program w pliku pod nazwą *Suma_12*.

Ćwiczenie 4. Obliczamy sumę *n* liczb wprowadzanych z klawiatury

1. Zmodyfikuj program zapisany w ćwiczeniu 3. tak, aby liczba wpłat była wprowadzana z klawiatury po uruchomieniu programu.
2. Zapisz program w pliku pod nazwą *Suma_n*.

2. Stosowanie instrukcji `while` w językach C++ i Python

Chcesz dalej oszczędzać na zakup nowego komputera, ponieważ potrzebna jest większa kwota. Zamierzasz wpłacać pieniądze, dopóki suma wpłat nie przekroczy potrzebnej kwoty. Jak przedstawić ten problem w wybranym języku programowania?

Nie wiemy z góry, ilu wpłat musimy dokonać, aby osiągnąć lub przekroczyć kwotę potrzebną na zakup komputera. Liczba iteracji zależy w tym przypadku od spełnienia warunku przekroczenia założonej kwoty.

Do zapisywania algorytmów iteracyjnych, w których liczba iteracji nie jest z góry określona, możemy zastosować następujące instrukcje iteracyjne:

- **`while`** – w języku C++ i w języku Python,
- **`do ... while`** – w języku C++.

Instrukcja iteracyjna **`while`**	**`while`** *(warunek)* *lista_instrukcji;* *kolejna_instrukcja;*	C++
	`while` *warunek:* *lista_instrukcji* *kolejna_instrukcja*	Python

***Tabela 3.** Ogólna postać instrukcji iteracyjnej* **`while`** *w językach C++ i Python*

Działanie pętli **`while`** jest takie samo w obydwu językach (C++ i Python): najpierw sprawdzany jest *warunek*; jeśli jest spełniony, to wykonywana jest *lista_instrukcji*. Wewnątrz bloku *lista_instrukcji* powinna być zawsze umieszczona instrukcja, która zmienia wartość *warunku* – w przeciwnym wypadku pętla nigdy się nie zakończy. W szczególnej sytuacji, gdy *warunek* od razu nie jest spełniony, *lista_instrukcji* w ogóle nie zostanie wykonana. *Lista_instrukcji* może zawierać jedną lub wiele instrukcji.

Przykład 1. Stosowanie instrukcji `while` w językach C++ i Python

Zadanie: Napisz program obliczający sumę wpłat aż do osiągnięcia lub przekroczenia założonej kwoty.

Dane: liczba rzeczywista dodatnia *kwota*, oznaczająca kwotę potrzebną na zakup komputera, ciąg dowolnych liczb rzeczywistych dodatnich *wplata*, oznaczających kolejne comiesięczne wpłaty.

Wynik: liczba rzeczywista dodatnia *suma_wplat*, oznaczająca wartość sumy wszystkich wpłat.

C++

Zbiorka.cpp

```
#include <iostream>
using namespace std;

int main()
{
    float kwota, wplata, suma_wplat;

    cout << "Podaj potrzebna kwota: ";
    cin >> kwota;
    suma_wplat = 0;
    while (suma_wplat < kwota)
    {
        cout << "Wprowadz wplate: ";
        cin >> wplata;
        suma_wplat = suma_wplat + wplata;
    }
    cout << "Suma wplat wynosi: " << suma_wplat << " zl";

    return 0;
}
```

Python

Zbiorka.py - C:/Podręcznik IP Klasa II/Moduł C/Temat C1_zrzuty/Zbiorka.py (3.7.2)

File Edit Format Run Options Window Help

```
kwota = float(input("Podaj potrzebną kwotę: "))
suma_wplat = 0

while suma_wplat < kwota:
    wplata = float(input("Wprowadź wpłatę: "))
    suma_wplat = suma_wplat + wplata

print("Suma wpłat:", suma_wplat, "zł")

input("\n\nAby zakończyć, naciśnij Enter")
```

Ln: 13 Col: 0

Ćwiczenie 5. Analizujemy, testujemy i modyfikujemy gotowy program

1. Przepisz wybrany program z przykładu 1. Zapisz program w pliku pod nazwą *Zbiorka*.
2. Uruchom i przetestuj program dla kilku różnych wartości zmiennych. Wyjaśnij znaczenie poszczególnych wierszy programu.
3. Zmodyfikuj program tak, aby na ekranie wyświetlała się również bieżąca suma, uzyskana po każdej wpłacie, a na koniec – różnica pomiędzy kwotą wpłaconą a założoną (nadpłata). Dodaj odpowiednie komunikaty. Zapisz plik pod tą samą nazwą.

Ćwiczenie 6. Sprawdzamy poprawność danych

1. Dodaj do programu zapisanego w ćwiczeniu 5. sprawdzanie poprawności wprowadzanych wartości zmiennych *kwota* i *wplata*, zgodnie ze specyfikacją zadania (podaną w przykładzie 1.). Wprowadzanie i sumowanie liczb powinny być wykonywane dla poprawnych wartości zmiennych *kwota* i *wplata*:
   a. jeśli użytkownik wprowadzi niepoprawną wartość zmiennej *kwota*, wyświetl na ekranie komunikat „Wprowadzono bledna liczbe" i zakończ program.
   b. jeśli użytkownik wprowadzi niepoprawną wartość zmiennej *wplata*, wyświetl na ekranie komunikat „Wprowadzono bledna liczbe" i zignoruj wprowadzoną wartość.
2. Zapisz plik pod tą samą nazwą. Uruchom program i sprawdź jego działanie dla różnych wartości zmiennych.

Ćwiczenie 7. Stosujemy instrukcję iteracyjną `while`

1. Napisz specyfikację zadania i program, który będzie obliczał sumę liczb całkowitych wprowadzanych z klawiatury aż do wprowadzenia zera. Wprowadzenie zera kończy zliczanie. Wynik sumowania wyprowadź na ekran.
2. Zapisz program w pliku pod nazwą *Suma_while*.

Wskazówki: Na początku programu przypisz zmiennej *liczba* wartość różną od zera, np. 1, a zmiennej *suma* – 0.

C++ Początkową wartość można przypisać zmiennej podczas jej deklaracji:

```
int liczba = 1, suma = 0;
```

Python

```
liczba = 1
suma = 0
```

3. Stosowanie instrukcji `do ... while` w języku C++

W języku C++ występuje instrukcja iteracyjna: **`do`** … **`while`**, którą możemy zastosować w przypadku, gdy nie znamy liczby iteracji.

Instrukcja iteracyjna **`do ... while`**	**`do`** *lista _ instrukcji;* **`while`***(warunek);* *kolejna_instrukcja;*	C++

Tabela 4. *Ogólna postać instrukcji iteracyjnej* **`do`** … **`while`** *w języku C++*

W instrukcji **`do`** … **`while`** najpierw jest wykonywana *lista_instrukcji*, a dopiero potem jest sprawdzany *warunek*.

Instrukcje w pętli (*lista_instrukcji*) są wykonywane, dopóki warunek jest prawdziwy. Jeśli warunek jest fałszywy – wykonywana jest *kolejna_ instrukcja*. Niezależnie od wartości początkowej warunku, *lista_instrukcji* zostanie wykonana przynajmniej raz. Jako *lista_instrukcji* może wystąpić pojedyncza instrukcja (w tym instrukcja **`do`** … **`while`**) lub więcej instrukcji ujętych w blok {}.

Przykład 2. Stosowanie instrukcji `do ... while` w języku C++

Zadanie: Napisz program zliczający znaki wprowadzane z klawiatury aż do wprowadzenia kropki. Wprowadzenie kropki kończy zliczanie. Wyprowadź na ekran odpowiedni komunikat i liczbę wprowadzonych znaków łącznie z kropką.

Dane: dowolny znak: *znak*.

Wynik: liczba całkowita *liczba_znakow*, oznaczająca liczbę znaków wprowadzonych z klawiatury.

Uwagi:

- Pojedynczy znak umieszczamy w programie w pojedynczych górnych cudzysłowach.
- Instrukcja `liczba_znakow++;` to odpowiednik instrukcji:
  `liczba_znakow = liczba_znakow + 1;`

C++

Zliczanie_znakow.cpp

```
#include <iostream>
using namespace std;

int main()
{
    char znak;
    int liczba_znakow = 0;

    do
    {
        cout << "Wprowadz znak: ";
        cin >> znak;
        liczba_znakow++;
    }
    while (znak != '.');
    cout << "Liczba wprowadzonych znakow wynosi: " << liczba_znakow;
    return 0;
}
```

Ćwiczenie 8. Analizujemy, testujemy i modyfikujemy gotowy program

1. Przepisz program z przykładu 2. Zapisz program w pliku pod nazwą *Zliczanie_znakow*.
2. Uruchom i przetestuj program dla kilku różnych wartości zmiennych. Wyjaśnij znaczenie poszczególnych wierszy programu.
3. Zmodyfikuj program tak, aby nie zliczał kropki. Zapisz program w pliku pod nazwą *Zliczanie_znakow_bez_kropki*.

Ćwiczenie 9. Stosujemy instrukcję iteracyjną `do ... while` w języku C++

1. Napisz specyfikację zadania i program, który będzie obliczał różnicę dwóch liczb całkowitych (*odjemna*, *odjemnik*) wprowadzanych z klawiatury, dopóki wynik odejmowania (*roznica*) będzie dodatni. Wynik niedodatni kończy obliczenia. Zliczaj wykonane działania odejmowania, których wynik jest dodatni, i wyprowadź ich liczbę na ekran.
2. Zapisz program w pliku pod nazwą *Roznice*.

Wskazówka: W warunku logicznym instrukcji `do ... while` sprawdzaj, czy wynik odejmowania (*roznica*) jest dodatni.

Warto zapamiętać

- W językach programowania występują instrukcje iteracyjne, które umożliwiają zapis algorytmów iteracyjnych, np.:
  - `for` – w językach C++ i Python,
  - `while` – w językach C++ i Python,
  - `do ... while` – w języku C++.

  Zwykle gdy znamy liczbę iteracji, stosujemy instrukcję `for`; jeśli nie wiemy, ile będzie powtórzeń – pozostałe dwie.
- W instrukcji `while` najpierw jest sprawdzany warunek logiczny, a potem (gdy warunek jest prawdziwy) są wykonywane instrukcje znajdujące się wewnątrz instrukcji `while`. Gdy warunek od razu nie jest spełniony, instrukcje wewnątrz instrukcji `while` nie zostaną wykonane ani razu.
- Instrukcja `do ... while` różni się od `while` tym, że warunek jest sprawdzany po bloku instrukcji umieszczonym pomiędzy słowami kluczowymi `do` i `while`, czyli te instrukcje zostaną wykonane przynajmniej raz.

Pytania i polecenia

1. Jakie instrukcje w językach programowania umożliwiają zapis algorytmów iteracyjnych?
2. Wyjaśnij na przykładzie wybranego języka programowania, jak działa instrukcja `while`.
3. Czy zostaną wykonane instrukcje wewnątrz instrukcji `while`, jeśli warunek jest prawdziwy? Uzasadnij odpowiedź.
4. Czym różni się działanie instrukcji `while` od `do ... while`?

Zadania

1. Do programu zapisanego w ćwiczeniu 4. dodaj sprawdzanie poprawności danych: wprowadzane kwoty i liczba wpłat powinny być liczbami dodatnimi (liczba wpłat jest liczbą naturalną). Zapisz plik pod tą samą nazwą.
2. Znajdź w Internecie wartość średniej temperatury w Polsce w poszczególnych miesiącach – w dzień i w nocy. Napisz program obliczający na podstawie tych danych średnie roczne temperatury w dzień i w nocy. Wyniki wyświetl na ekranie. Zapisz program w pliku pod nazwą *Temperatury*.
3. Napisz program obliczający i wyświetlający na ekranie iloraz dwóch liczb całkowitych (*dzielna*, *dzielnik*) wprowadzanych z klawiatury aż do wprowadzenia dzielnika równego zero. Ponieważ nie da się obliczyć ilorazu dla dzielnika równego zero, program powinien wyświetlić odpowiedni komunikat. Zapisz program w pliku pod nazwą *Iloraz*.
4. Napisz program obliczający i wyświetlający na ekranie objętość sześcianu o boku *bok* aż do wprowadzenia niedodatniej wartości zmiennej *bok*. Zapisz program w pliku pod nazwą *Objetosc_szescianu*.
5. Napisz specyfikację zadania i program zliczający, ile jest liczb podzielnych przez 3 pośród liczb całkowitych wprowadzanych z klawiatury. W przypadku wprowadzenia zera program ma zakończyć działanie. Zapisz program w pliku pod nazwą *Podzielne_przez_3*.

Dla zainteresowanych

6. Masz *n* zł na kupienie upominków dla znajomych. Upominki możesz kupować, dopóki starczy ci pieniędzy. Ceny upominków wprowadzaj z klawiatury. Nie możesz przekroczyć kwoty *n* zł ani wydać mniej. Jeśli cena ostatniego upominku przekroczy kwotę *n*, musisz kupić coś tańszego. Program kończy się, gdy wydasz dokładnie *n* zł. Napisz odpowiedni program i zapisz go w pliku pod nazwą *Upominki*.
7. Napisz program obliczający część całkowitą pierwiastka kwadratowego z podanej liczby naturalnej *n*.
   Wskazówka: Istnieją różne algorytmy obliczania pierwiastka. Można na przykład obliczać kwadraty kolejnych liczb naturalnych aż do uzyskania liczby większej lub równej *n*.

Przeczytaj jeśli chcesz wiedzieć więcej…

W języku Python można przypisać wiele wartości wielu zmiennym, umieszczając nazwy zmiennych i ich wartości w jednej instrukcji przypisania, np.:

```
liczba, suma = 1, 0
```

co odpowiada instrukcjom:

```
liczba = 1
suma = 0.
```

W języku Python instrukcja **`while`** może występować ze słowem kluczowym **`else`**. Jest ona jednak używana bardzo rzadko. W przypadku użycia **`else`** *kolejna_instrukcja* zostanie wykonana po uprzednim wykonaniu *listy _instrukcji2*.

Instrukcja iteracyjna **`while`** z klauzulą **`else`**	**`while`** *warunek:*
	  lista_instrukcji1
	`else`*:*
	  lista_instrukcji2
	kolejna_instrukcja
		Python

Tabela 5. *Ogólna postać instrukcji iteracyjnej* **`while`** *z klauzulą else w języku Python*