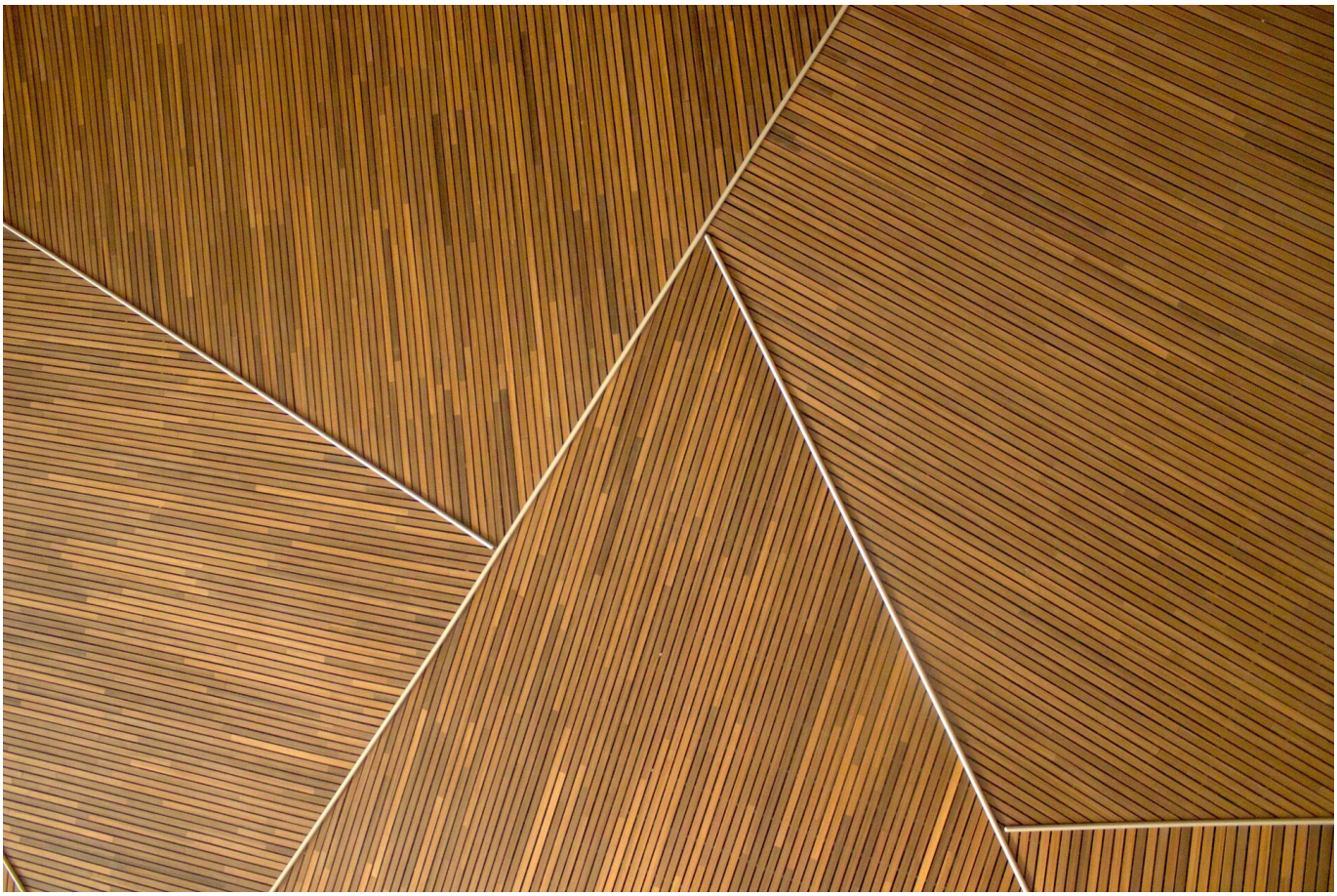
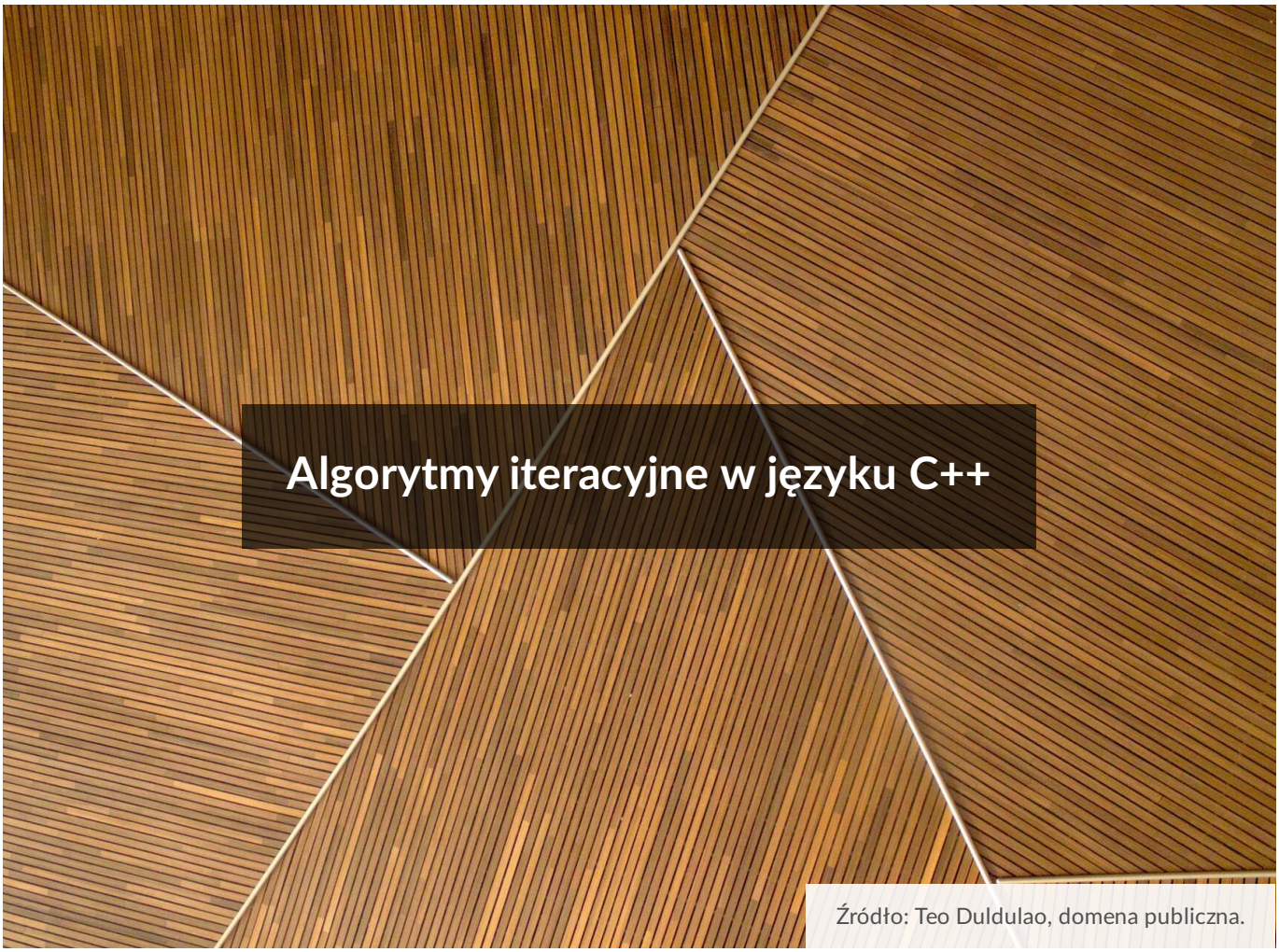


Algorytmy iteracyjne w języku C++

- Wprowadzenie
- Schemat interaktywny
- Przeczytaj
- Sprawdź się
- Dla nauczyciela



Źródło: Teo Duldulao, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Jak już wiemy z e-materiału Algorytmy iteracyjne, iteracja to powtarzanie zapisanych w programie instrukcji w pętli z góry określoną liczbę razy lub do momentu spełnienia określonego warunku. Mechanizm ten jest podstawową operacją wykorzystywaną w programowaniu.

Tym razem zapoznamy się z algorytmami iteracyjnymi w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Algorytmy iteracyjne w języku Java,
- Algorytmy iteracyjne w języku Python.

Więcej zadań? Sięgnij do Algorytmy iteracyjne – zadania maturalne.

> **Twoje cele**
>
> - Scharakteryzujesz kilka pętli używanych w języku C++, takich jak `while`, `do-while` oraz `for`.
> - Wyjaśnisz, jak za pomocą pętli stworzyć algorytmy iteracyjne w języku C++.

- Rozwiążesz kilka zadań z zastosowaniem iteracji.

Schemat interaktywny

Polecenie 1

Stwórz program rysujący tabliczkę mnożenia o rozmiarze `rozmiar x rozmiar`.

Przykładowa tabliczka mnożenia, jeśli `rozmiar = 3`:

```
  1  2  3
1 1  2  3
2 2  4  6
3 3  6  9
```

Specyfikacja problemu:

Dane:

- `rozmiar` – liczba naturalna oznaczająca liczbę wierszy i kolumn tworzących tabliczkę o rozmiarze `rozmiar x rozmiar`

Wynik:

Na standardowym wyjściu program wypisuje tabliczkę mnożenia zbudowaną z `rozmiar` wierszy i `rozmiar` kolumn.

Polecenie 2
Wykorzystując schemat interaktywny, stwórz algorytm gry FizzBuzz dla liczb z przedziału $\langle 1, \texttt{limit} \rangle$. Przetestuj działanie programu dla przedziału $\langle 1, 20 \rangle$.

> **Ważne!**
>
> Program powinien zastąpić liczby podzielne przez 3 słowem „Fizz”, liczby podzielne przez 5 słowem „Buzz”, natomiast liczby podzielne przez 3 i 5 słowem „FizzBuzz”.

Specyfikacja problemu:

Dane:

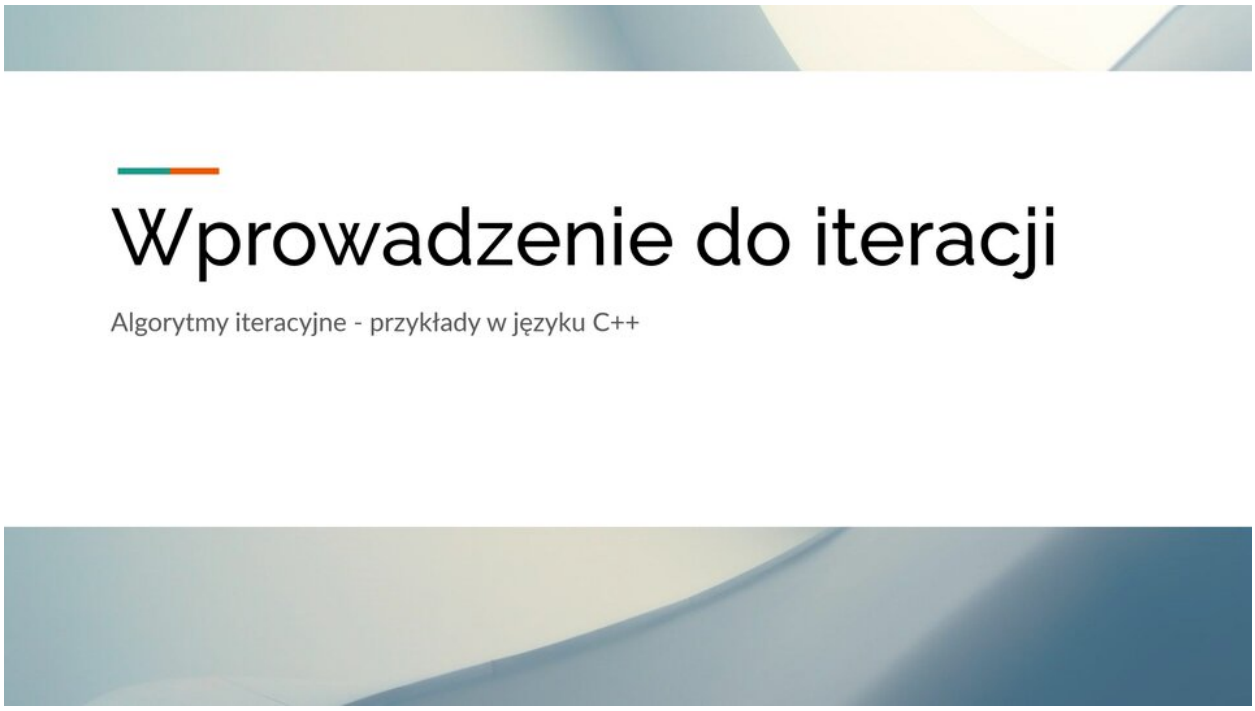
- `limit` – liczba naturalna, na której program powinien zakończyć działanie

Wynik:

Na standardowym wyjściu program wypisuje kolejne liczby od 1 do `limit`, zastępując liczby podzielne przez 3 słowem „Fizz”, liczby podzielne przez 5 słowem „Buzz”, natomiast liczby podzielne przez 3 oraz przez 5 słowem „FizzBuzz”.

Polecenie 3
Porównaj swoje rozwiązania z filmem.

Wystąpił błąd



Film dostępny pod adresem /preview/resource/RKnteTq6eXgVv

Film nawiązujący do treści lekcji dotyczący algorytmów iteracyjnych na przykładzie języka C++.

Przeczytaj

Algorytm iteracyjny w C++

Zanim napiszesz swój pierwszy program realizujący algorytm iteracyjny w języku C++, musisz się dowiedzieć, jak w takim programie zastosować pętle. Pętla pozwala na wykonanie sekwencji instrukcji w programie określoną liczbę razy. Wyobraźmy sobie, że chcemy wypisać na ekranie wszystkie liczby z zakresu od 1 do 50 i dla każdej z nich wyświetlić komunikat, czy jest ona liczbą podzielną przez 4. Pięćdziesięciokrotne zapisanie ciągu tych samych poleceń byłoby bardzo żmudne i czasochłonne, a kod długi i niezbyt czytelny. Dlatego podczas pisania programu skorzystamy z instrukcji pętli, które znacząco przyspieszą i ułatwią pracę. Jakie instrukcje iteracyjne mamy zatem do dyspozycji w języku C++?

> **Ważne!**
>
> W języku C++ możemy korzystać z następujących instrukcji iteracyjnych:
>
> - `while`
> - `do-while`
> - `for`

Pętla `while`

Zacznijmy od omówienia pętli `while`. Jej składnia w języku C++ wygląda następująco:

```
while (warunekWykonaniaPętli) {
    // ciąg instrukcji
}
```

Na początku pętli musi pojawić się słowo kluczowe `while`. W przypadku spełnienia warunku `warunekWykonaniaPętli` zostanie wykonany ciąg instrukcji wewnątrz nawiasów klamrowych. Instrukcje będą cyklicznie wykonywane do momentu, gdy warunek pętli przestanie być spełniony. Jeżeli już na początku warunek nie będzie spełniony, polecenia w pętli nie zostaną ani razu wykonane.

Algorytm iteracyjny z wykorzystaniem instrukcji `while`

Zobaczmy teraz, jak będzie wyglądała pętla dla konkretnego problemu. Napiszmy zatem część programu realizującego algorytm zadany na początku tej sekcji niniejszego

e-materiału. Niech zostaną wypisane wszystkie liczby z zakresu od 1 do 50 z odpowiednim komunikatem o podzielności liczby przez 4.

```
#include <iostream>

int main() {
    int i = 1;

    while (i <= 50) {
        std::cout << i << std::endl;

        if ((i % 4) == 0) {
            std::cout << "Liczba " << i <<  " jest podzielna prze
        } else {
            std::cout << "Liczba " << i <<  " nie jest podzielna
        }

        i++;
    }

    return 0;
}
```

Algorytm ten jest algorytmem iteracyjnym. Na początku deklarujemy zmienną całkowitą `i`, w której zapiszemy wartość 1. Dopóki wartość zapisana w `i` będzie mniejsza lub równa 50, program będzie wypisywał ją na ekranie – dzięki temu zostaną wypisane wszystkie liczby z zakresu 1 do 50. Następnie sprawdzamy, czy reszta z dzielenia danej liczby przez 4 jest równa 0. Jeśli tak, oznacza to, że jest podzielna przez 4, wyświetlamy zatem odpowiedni komunikat. W przeciwnym razie wypisujemy, że liczba **nie jest** podzielna przez 4. Na koniec w pętli inkrementujemy zmienną `i`, aby móc poddać analizie kolejną liczbę.

Pętla `do-while`

Przedstawmy teraz składnię pętli `do-while` w języku C++:

```
do {
    // ciąg instrukcji
} while (warunekWykonaniaPętli);
```

W przypadku tej instrukcji pętla rozpoczyna się słowem kluczowym `do`, a kończy się `while`. Ponownie wykonywany jest ciąg instrukcji zawarty w nawiasach klamrowych, gdy spełniony zostaje `warunekWykonaniaPętli`. Tym razem jednak jest on sprawdzany na końcu pętli. Oznacza to, że mamy pewność, iż instrukcje zostaną wykonane co najmniej jeden raz.

Algorytm iteracyjny z wykorzystaniem instrukcji `do-while`

Sprawdźmy instrukcję `do-while` w praktyce. Napiszmy algorytm iteracyjny, który wypisze liczby od 35 do 12, a na końcu wyświetli sumę wypisanych liczb.

```
#include <iostream>

int main() {
    int i = 35;
    int suma = 0;

    do {
        std::cout << i << std::endl;
        suma += i;
        i--;
    } while (i >= 12);

    std::cout << "Suma wszystkich liczb wynosi: " << suma;
    return 0;
}
```

W powyższym algorytmie deklarujemy zmienną, której nadajemy wartość 35. W tym przypadku zmienna `i` będzie zmniejszana, ponieważ chcemy wyświetlać liczby malejąco. Deklarujemy także zmienną `suma`, w której będą przechowywane aktualne wartości sumy wyświetlanych liczb.

Dzięki temu, że warunek w pętli `do-while` sprawdzany jest na końcu, od razu wykonywane są instrukcje w nawiasach klamrowych. Wewnątrz pętli wyświetlamy aktualną wartość zmiennej `i`, po czym dodajemy tę wartość do zmiennej `suma`. Następnie dekrementujemy zmienną `i`. Na koniec sprawdzany jest warunek pozwalający na przejście do kolejnej iteracji.

Pętla `for`

Składnia pętli `for` wygląda następująco:

```
for (warunekPoczątkowy; warunekWykonywania; modyfikacja) {
    // ciąg instrukcji
}
```

Pętla `for` zawiera na początku słowo kluczowe `for`. Z kolei `warunekPoczątkowy` służy do stworzenia potrzebnych zmiennych, np. iteratora. Następnie, jeżeli `warunekWykonania` jest spełniony, zostaje wykonywany ciąg instrukcji wewnątrz pętli. Ostatni element, czyli `modyfikacja`, jest instrukcją zmieniającą wartość wyrażenia inicjującego po każdej iteracji pętli.

Algorytm iteracyjny z wykorzystaniem instrukcji `for`

Napiszmy algorytm iteracyjny obliczający wartość 4! z wykorzystaniem pętli `for`.

```
#include <iostream>

int main() {
    int podstawa = 4;
    int silnia = 1;

    if (podstawa != 0) {
        for (int i = 1; i <= podstawa; i++) {
            silnia *= i;
        }
    }

    std::cout << silnia;
    return 0;
}
```

W algorytmie deklarujemy zmienną `podstawa` i przypisujemy do niej liczbę, z której chcemy obliczyć silnię. Z kolei w zmiennej `silnia` będzie zapisywany wynik obliczenia silni.

- **Wyrażeniem inicjującym** jest zmienna `i`, której przypisujemy wartość 1.
- **Warunkiem wykonania pętli** jest sprawdzenie, czy wyrażenie inicjujące jest mniejsze lub równe 4.
- **Wyrażeniem modyfikującym** jest inkrementacja zmiennej `i`.

4! jest iloczynem liczb naturalnych mniejszych lub równych 4, czyli:

$$4! = 1 \cdot 2 \cdot 3 \cdot 4$$

W związku z powyższym, zadaniem pętli będzie mnożenie wyniku `silnia` przez kolejne liczby naturalne: 1, 2, 3, 4. Oto polecenia wykonywane w ramach pętli `for`:

1. Zaczynamy od wyrażenia inicjującego - zmiennej `i` przypisywana jest wartość 1.
2. Sprawdzamy, czy zmienna `i` jest mniejsza lub równa zmiennej `podstawa`. Jeżeli tak, przechodzimy do kolejnego kroku; jeżeli nie – wychodzimy z pętli.
3. Następnie wykonywane są instrukcje wewnątrz pętli.
4. Zwiększona zostaje zmienna `i` w celu uzyskania kolejnej liczby.
5. Następuje powrót do kroku 2.

Słownik

dekrementacja

zmniejszenie wartości argumentu o 1
inkrementacja

zwiększenie wartości argumentu o 1
pętla

instrukcja iteracyjna, która umożliwia cykliczne wykonywanie ciągu instrukcji
silnia

$n!$ - iloczyn liczb naturalnych mniejszych lub równych n

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który będzie obliczał iteracyjnie wartość potęgi. Przetestuj jego działanie dla potęgi o podstawie równej 7 i wykładniku równym 13.

Specyfikacja problemu:

Dane:

- `podstawa` – liczba naturalna = 7, która zostanie podniesiona do zadanej potęgi
- `wykladnik` - liczba naturalna oznaczająca liczbę czynników w mnożeniu

Wynik:

Na standardowym wyjściu prezentowany jest wynik potęgowania.

Ćwiczenie 2

Napisz program, który będzie obliczał iteracyjnie wartość silni. Przetestuj jego działanie dla 11!.

Specyfikacja problemu:

Dane:

- `podstawa` – liczba naturalna

Wynik:

Na standardowym wyjściu prezentowana jest silnia zadanej liczby.

Ćwiczenie 3

Napisz program, który będzie obliczał iteracyjnie wartość n-tego elementu ciągu Fibonacciego. Ciąg Fibonacciego to ciąg liczb naturalnych, w którym pierwsze dwa wyrazy są równe 1, a każdy kolejny wyraz jest sumą dwóch poprzednich. Przetestuj działanie swojego programu dla 40. elementu ciągu Fibonacciego.

Specyfikacja problemu:

Dane:

- n – indeks danego elementu ciągu Fibonacciego; liczba naturalna

Wynik:

Na standardowym wyjściu prezentowana jest n-ty element ciągu Fibonacciego.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Algorytmy iteracyjne w języku C++

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> Zakres podstawowy. Uczeń:
>
> 1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);
>
> Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
>
> 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;
>
> 3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Scharakteryzujesz kilka pętli używanych w języku C++, takich jak `while`, `do-while` oraz `for`.
- Wyjaśnisz, jak za pomocą pętli stworzyć algorytmy iteracyjne w języku C++.
- Rozwiążesz kilka zadań z zastosowaniem iteracji.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Algorytmy iteracyjne w języku C++”. Uczniowie zapoznają się z multimedium w sekcji „Schemat interaktywny”.

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat, wskazuje cele zajęć oraz ustala z uczestnikami zajęć kryteria sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj" wyświetlone na tablicy.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny", czyta treść polecenia nr 1 i omawia kolejne kroki rozwiązania. W kolejnym kroku uczniowie w parach wykonują polecenie nr 2 i porównują swoje rozwiązanie z przedstawionym w filmie.
3. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1 i 2 z sekcji „Sprawdź się" , po ich wykonaniu porównują otrzymane wyniki z inną osobą.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku C++.

Praca domowa:

1. Uczniowie wykonują ćwiczenie 3 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać multimedia z sekcji: „Schemat interaktywny", „Przeczytaj", „Sprawdź się" do przygotowania się do lekcji powtórkowej.