Szyfr Playfair w języku C++

Szyfr Playfair w języku C++

Źródło: Diomari Madulara, domena publiczna.

Szyfr Playfair, zwany także szyfrem Playfaira (od nazwiska jego twórcy), to jeden z popularniejszych szyfrów podstawieniowych, czyli polegających na zastępowaniu poszczególnych znaków innymi. W przeciwieństwie jednak do szyfru Cezara jest on poligramowy - nie zamieniamy tu pojedynczych znaków, lecz ich pary. Wprowadza to więc dodatkową warstwę obliczeń, które muszą zostać wykonane, aby zakodować lub odkodować dowolną treść za pomocą szyfru Playfair. W tym e-materiale dowiesz się, jak zaimplementować go w języku C++.

Podstawowe informacje na temat tego szyfru znajdziesz w e-materiale Szyfr Playfair.

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

- Szyfr Playfair w języku Java,
- Szyfr Playfair w języku Python.

Więcej zadań? Przejdź do Szyfr Playfair – zadania maturalne.

Twoje cele

- Przedstawisz techniczną stronę działania szyfru Playfair.

- Zaimplementujesz program służący do szyfrowania tekstu przy użyciu bibliotek standardowych języka C++.
- Przeanalizujesz sposób prostego deszyfrowania zakodowanego tekstu bez konieczności wykorzystywania dodatkowych funkcji.

Przeczytaj

Problem 1

Zaimplementuj szyfr Playfair. Przetestuj jego działanie dla klucza `JESIEŃ` oraz wiadomości `INSTRUMENT`. Upewnij się, że program poprawnie deszyfruje szyfrogram.

Specyfikacja:

Dane:

- `klucz` – łańcuch znaków
- `wiadomosc` – łańcuch znaków

Wynik:

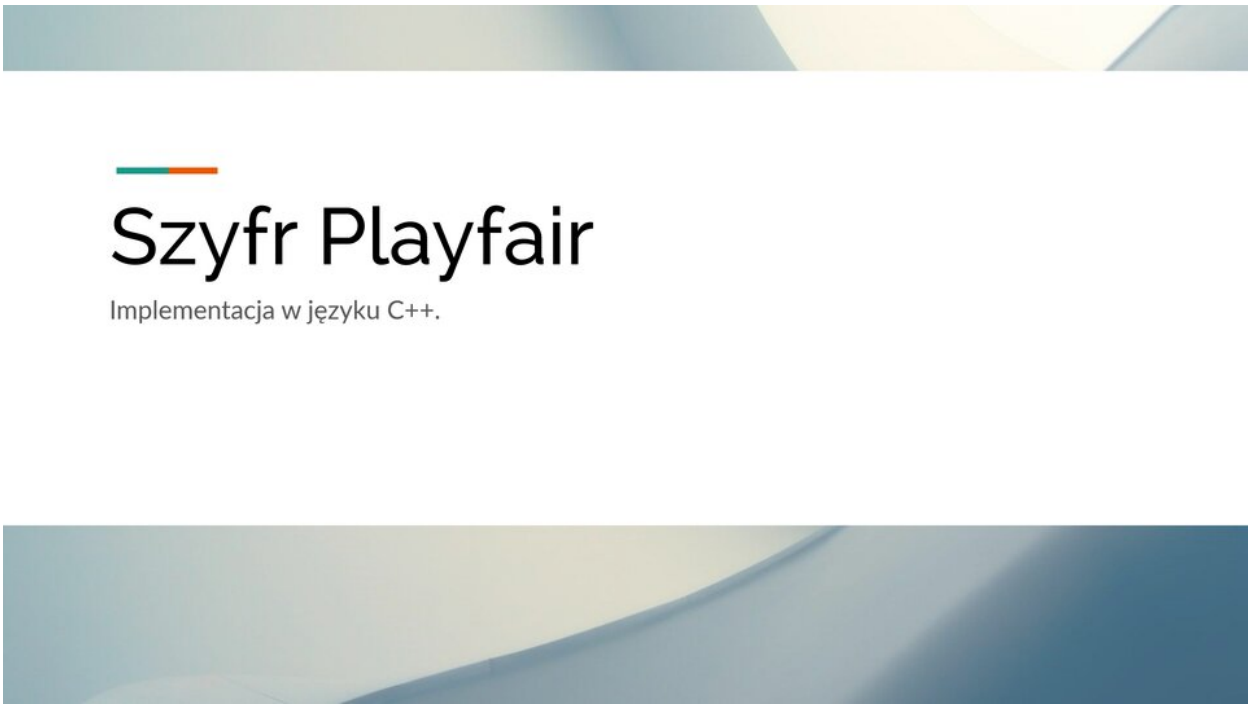
Program na wyjściu standardowym zwróci zaszyfrowany za pomocą klucza `klucz` łańcuch znaków `wiadomosc`. Następnie zwróci ten sam łańcuch znaków po deszyfrowaniu.

```
#include <iostream>

using namespace std;

int main () {

    // Tutaj dodaj kod. Żeby coś wypisać, użyj polecenia: cout
}
```

```
```

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.



Film dostępny pod adresem /preview/resource/RpJC1dfVOUgN4

Film nawiązujący do treści materiału - dotyczy szyfru Playfair.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 5.29 KB w języku polskim

Podsumowanie

Szyfrowanie par

Dla każdej pary liter w tekście, jaki musimy zaszyfrować, należy określić ich względne położenie. Jeżeli tekst ma nieparzystą liczbę liter, dodajmy na jego końcu literę Z – jest to często stosowana metoda.

Interesują nas trzy możliwe sytuacje:

- litery znajdują się **w tym samym wierszu** – wtedy przesuwamy je o 1 jednostkę w prawo, jeżeli to potrzebne, wracając do początku wiersza (nie zmieniając kolumny!). Przykładowo, dla pary MB dla naszego klucza wynikiem jest BO;
- litery znajdują się **w tej samej kolumnie** – wtedy przesuwamy je o 1 jednostkę w dół, jeżeli to potrzebne, wracając do początku kolumny (nie zmieniając wiersza!). Przykładowo, dla pary PX dla naszego klucza wynikiem jest XR;
- litery znajdują się **na różnych wierszach i kolumnach** – w tym przypadku traktujemy te litery jako końce przekątnej prostokąta. Wierzchołki drugiej przekątnej tego prostokąta będą zaszyfrowaną parą. Przykładowo, dla pary SL dla naszego klucza wynikiem jest JY.

Tworzenie klucza

Zacznijmy od napisania funkcji, która stworzy klucz na podstawie pewnego słowa bazowego. Będzie ona przyjmowała i zwracała po jednym argumencie typu `string` – odpowiednio słowo bazowe oraz gotowy klucz. Potraktujemy tabelę 5 na 5 jako jeden ciąg znaków.

> **Ważne!**
>
> W programie będziemy używali dwóch bibliotek standardowych: `iostream` i `string` – pamiętajmy o ich dołączeniu na początku kodu. Nie zapominajmy również o przestrzeni nazw `using namespace std;`.

Na początek zapiszmy funkcję:

```
1 // Funkcja do stworzenia tabeli na podstawie podanego klucza
2 string przetwarzajKlucz(string klucz) {
3
4 }
```

Stwórzmy alfabet, którego będziemy używali w kluczu. Będą to wielkie litery alfabetu łacińskiego z pominięciem litery Q.

```
// Funkcja do stworzenia tabeli na podstawie podanego klucza
string przetwarzajKlucz(string klucz) {
    /*
    Stwórzmy używany przez nas alfabet; duże litery
    alfabetu łacińskiego, bez litery Q
    */
    string alfabet;
    for (char i = 'A'; i <= 'Z'; i++)
        if (i != 'Q')
            alfabet += i;

}
```

Upewnijmy się, że wszystkie litery słowa bazowego są wielkimi literami. Skorzystamy w tym celu z kolejnej pętli.

```
// Funkcja do stworzenia tabeli na podstawie podanego klucza
string przetwarzajKlucz(string klucz) {
    /*
    Stwórzmy używany przez nas alfabet; duże litery
    alfabetu łacińskiego, bez litery Q
    */
    string alfabet;
    for (char i = 'A'; i <= 'Z'; i++)
        if (i != 'Q')
            alfabet += i;

    for (auto& x : klucz)
        x = toupper(x);
}
```

Utwórzmy zmienną, w której będziemy przechowywali wyjściowy klucz. Dodajmy do niego wszystkie znaki – bez powtórzeń – jakie mamy w słowie bazowym.

```
// Funkcja do stworzenia tabeli na podstawie podanego klucza
string przetwarzajKlucz(string klucz) {
    /*
    Stwórzmy używany przez nas alfabet; duże litery
    alfabetu łacińskiego, bez litery Q
    */
```

```
    string alfabet;
    for (char i = 'A'; i <= 'Z'; i++)
        if (i != 'Q')
            alfabet += i;

    for (auto& x : klucz)
        x = toupper(x);

    string przetwarzanyKlucz;
    /*
    Dodajmy do klucza wyjściowego wszystkie znaki,
    bez powtórzeń, jakie mamy w podanym wyrażeniu
    */
    for (auto x : klucz)
        if (alfabet.find(x) != string::npos &&
                przetwarzanyKlucz.find(x) == string::npos)
            przetwarzanyKlucz += x;
```

Pozostaje tylko dopełnić tabelę pozostałymi literami alfabetu, a następnie zwrócić ostateczny klucz.

```
// Funkcja do stworzenia tabeli na podstawie podanego klucza
string przetwarzajKlucz(string klucz) {
    /*
    Stwórzmy używany przez nas alfabet; duże litery
    alfabetu łacińskiego, bez litery Q
    */
    string alfabet;
    for (char i = 'A'; i <= 'Z'; i++)
        if (i != 'Q')
            alfabet += i;

    for (auto& x : klucz)
        x = toupper(x);

    string przetwarzanyKlucz;
    /*
    Dodajmy do klucza wyjściowego wszystkie znaki,
    bez powtórzeń, jakie mamy w podanym wyrażeniu
    */
    for (auto x : klucz)
```

```
        if (alfabet.find(x) != string::npos &&
                przetwarzanyKlucz.find(x) == string::npos)
            przetwarzanyKlucz += x;

    // Teraz dopełnijmy klucz pozostałymi literami alfabetu
    for (auto x : alfabet)
        if (przetwarzanyKlucz.find(x) == string::npos)
            przetwarzanyKlucz += x;

    return przetwarzanyKlucz;
}
```

Funkcja szyfrująca

Napiszemy funkcję, która będzie implementowała sam algorytm szyfrowania. Będzie ona przyjmowała tekst do zaszyfrowania oraz klucz, a zwracała zaszyfrowany tekst – wszystkie wartości są oczywiście typu `string`. Po upewnieniu się, że tekst ma parzystą liczbę znaków, przechodzimy po wszystkich parach liter i zamieniamy je na odpowiednie wartości z klucza, korzystając z omówionych wcześniej zasad.

Zacznijmy od stworzenia funkcji kodującej podany tekst za pomocą klucza.

```
// Funkcja do zakodowania wiadomości za pomocą klucza
string szyfruj(string wiadomosc, const string klucz) {

}
```

Jeżeli tekst ma nieparzystą liczbę znaków, dodajmy na końcu literę Z. Następnie stwórzmy pętlę, która przejdzie po wszystkich parach znaków w tekście.

```
// Funkcja do zakodowania wiadomości za pomocą klucza
string szyfruj(string wiadomosc, const string klucz) { {
    /*
    Upewnijmy się, że tekst do zaszyfrowania
    ma parzystą liczbę znaków
    */
    if (wiadomosc.length() % 2) {
        wiadomosc += 'Z';
    }

```

```
    // Interesują nas pary liczb, dlatego i+=2
    for (unsigned int i = 0; i < wiadomosc.length(); i += 2) {

    }
}
```

Znajdźmy pozycje obu liter z pary w kluczu.

```
// Funkcja do zakodowania wiadomości za pomocą klucza
string szyfruj(string wiadomosc, const string klucz) { {
    /*
    Upewnijmy się, że tekst do zaszyfrowania
    ma parzystą liczbę znaków
    */
    if (wiadomosc.length() % 2) {
        wiadomosc += 'Z';
    }

    // Interesują nas pary liczb, dlatego i+=2
    for (unsigned int i = 0; i < wiadomosc.length(); i += 2) {
        /*
        Znajdźmy wiersze i kolumny rozpatrywanych
        liter w naszym kluczu
        */
        int row[2] = {
            klucz.find(wiadomosc[i]) / 5,
            klucz.find(wiadomosc[i + 1])/ 5
        };
        int col[2] = {
            klucz.find(wiadomosc[i]) % 5,
            klucz.find(wiadomosc[i + 1]) % 5
        };
    }
}
```

Teraz wystarczy zapisać za pomocą kodu omówione wcześniej zasady zamieniania znaków. Na koniec funkcji zwracamy zaszyfrowany tekst.

```
// Funkcja do zakodowania wiadomości za pomocą klucza
string szyfruj(string wiadomosc, const string klucz) { {
```

```
    /*
    Upewnijmy się, że tekst do zaszyfrowania
    ma parzystą liczbę znaków
    */
    if (wiadomosc.length() % 2) {
        wiadomosc += 'Z';
    }

    // Interesują nas pary liczb, dlatego i+=2
    for (unsigned int i = 0; i < wiadomosc.length(); i += 2) {
        /*
        Znajdźmy wiersze i kolumny rozpatrywanych
        liter w naszym kluczu
        */
        int row[2] = {
            klucz.find(wiadomosc[i]) / 5,
            klucz.find(wiadomosc[i + 1])/ 5
        };
        int col[2] = {
            klucz.find(wiadomosc[i]) % 5,
            klucz.find(wiadomosc[i + 1]) % 5
        };

        // Dla dwóch liter w tej samej kolumnie:
        // przesuwamy je o 1 w dół
        if (col[0] == col[1]) {
            wiadomosc[i] = klucz[((row[0]+1)%5)*5 + col[0]];
            wiadomosc[i + 1] = klucz[((row[1]+1)%5)*5 + col[1]];
        }
        // Dla dwóch liter w tym samym rzędzie:
        // przesuwamy je o 1 w prawo
        else if (row[0] == row[1]) {
            wiadomosc[i] = klucz[row[0]*5 + (col[0]+1)%5];
            wiadomosc[i + 1] = klucz[row[1]*5 + (col[1]+1)%5];
        }
        // Dla liter w różnych rzędach i kolumnach:
        // bierzemy litery "na krzyż"
        else {
            wiadomosc[i] = klucz[row[1]*5 + col[0]];
            wiadomosc[i + 1] = klucz[row[0]*5 + col[1]];
        }
    }
```

```
    return wiadomosc;
}
```

Funkcja deszyfrująca

Moglibyśmy stworzyć funkcję deszyfrującą poprzez zmianę kierunku przesuwania wartości w prawo i w dół na odpowiednie przesuwanie ich w lewo i w górę. Jednak w takim wypadku znaczną część kodu skopiowalibyśmy bez potrzeby – jest to przykład negatywnego zjawiska zwanego redundancją.

Zaletą szyfru Playfair jest, że funkcję deszyfrującą możemy napisać za pomocą funkcji szyfrującej. Wystarczy, że przekażemy klucz odbity symetrycznie względem środka tabeli. Wtedy przesunięcia w prawo i w dół na naszym przekształconym kluczu będą odpowiadały odpowiednio przesunięciom w lewo i w górę.

Właśnie w tym celu przechowujemy klucz w zmiennej typu `string`. Efekt symetrii uzyskamy poprzez zwykłe podanie klucza od tyłu. Implementacja funkcji jest zatem krótka i zwięzła:

```
// Funkcja do odkodowania zaszyfrowanej wiadomości
string decode(string wiadomosc, const string klucz) {
    return szyfruj(wiadomosc, string(klucz.rbegin(), klucz.rend()
}
```

Wygląda to znacznie czytelniej niż kopiowanie poprzedniego kodu z niewielkimi zmianami.

Przykładowe użycie

Przykładowy kod demonstrujący użycie napisanych przez nas funkcji przedstawia się w następująco:

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    cout << "Podaj klucz do szyfru:" << endl;
    string klucz;
    getline(cin, klucz);
    klucz = przetwarzajKlucz(klucz);
```

```
11
12      cout << "Podaj tekst do zaszyfrowania:" << endl;
13      string wiadomosc;
14      cin >> wiadomosc;
15      // Zmieńmy wszystkie litery naszej wiadomości na wielkie
16      for (auto& x : wiadomosc)
17          x = toupper(x);
18
19      cout << "Do szyfrowania zostanie uzyty nastepujacy klucz:" <<
20      for (int i = 0; i < 25; i++) {
21          cout << klucz[i];
22          if ((i+1) % 5 == 0)
23              cout << endl;
24      }
25      cout << endl;
26
27      cout << "Zaszyfrowany tekst to:" << endl;
28      cout << (wiadomosc = szyfruj(wiadomosc, klucz)) << endl << en
29
30      cout << "Po przetworzeniu przez algorytm odszyfrowania otrzym
31      cout << (decode(wiadomosc, klucz)) << endl;
32
33      return 0;
34  }
```

Słownik

klucz

informacja służąca zazwyczaj do szyfrowania i deszyfrowania wiadomości

redundancja

inaczej nadmiarowość, czyli sytuacja, gdzie kod jest skopiowany w kilku różnych miejscach, podczas gdy może zostać użyta inna konstrukcja, na przykład funkcja czy pętla

szyfr podstawieniowy poligramowy

rodzaj szyfru podstawieniowego polegający na zamianie nie pojedynczych znaków, a całych ich grup na inne

tekst jawny

informacja przeznaczona do zaszyfrowania lub będąca rezultatem procesu deszyfrowania

Animacja

Polecenie 1

Zapoznaj się z animacją. Następnie wykonaj polecenie 2.

Film dostępny pod adresem /preview/resource/Rvpk4RS5uh77w

Film nawiązujący do szyfru Playfaira.

Polecenie 2

Poszukaj informacji o tym, jak wykorzystywano szyfr Playfair, zwany również szyfrem Playfaira.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Uzupełnij podaną funkcję `konwertuj` tak, aby zwracała ona tekst jawny dla zadanego, zaszyfrowanego szyfrem Playfair ciągu znaków oraz odpowiadającego mu klucza. Do stworzenia kwadratu szyfrującego wykorzystaj gotową funkcję `przetwarzajKlucz`.

Specyfikacja:

Dane:

- `wiadomosc` — zaszyfrowane szyfrem Playfair słowo przekazane do procesu odszyfrowywania; ciąg parzystej liczby znaków składający się z wielkich liter alfabetu angielskiego.

- `klucz` — klucz wykorzystywany w procesie odszyfrowywania; ciąg znaków składający się z liter alfabetu angielskiego.

Wynik:

- `jawny` — odszyfrowane słowo; ciąg znaków składający się z wielkich liter alfabetu angielskiego

Działanie swojego programu przetestuj dla szyfrogramu „BRDOWJZTKWMP” oraz klucza „wskaznik”.

Twoje zadania

1. Funkcja `konwertuj` zwraca tekst jawny dla podanego szyfrogramu oraz klucza.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Szyfr Playfair w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

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

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

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):

- Przedstawisz techniczną stronę działania szyfru Playfair.
- Zaimplementujesz program służący do szyfrowania tekstu przy użyciu bibliotek standardowych języka C++.
- Przeanalizujesz sposób prostego deszyfrowania zakodowanego tekstu bez konieczności wykorzystywania dodatkowych funkcji.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

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: „Szyfr Playfair w języku C++”. Uczniowie zapoznają się z treściami w sekcji „Przeczytaj” w kontekście programowania.

Faza wstępna:

1. Nauczyciel wyświetla uczniom temat zajęć oraz cele. Prosi, by na ich podstawie uczniowie sformułowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. Uczniowie zapoznają się z treścią problemu 1 z sekcji „Przeczytaj". Indywidualnie opracowują rozwiązanie, a następnie porównują je z zaprezentowanym na filmie.
2. **Praca z multimedium.** Nauczyciel czyta polecenie 1 z sekcji „Animacja". Uczniowie w parach piszą program, następnie porównują swoje rozwiązania z przedstawionymi w filmie.
3. **Ćwiczenia umiejętności.** Uczniowie w parach wykonują ćwiczenie 1 z sekcji „Sprawdź się". Następnie porównują swoje rozwiązania z inną grupą.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie dodają do swoich rozwiązań z sekcji „Sprawdź się" komentarze tak, by nawet osoba, która nie zna programowania, mogła zrozumieć zachodzące procesy.

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:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.