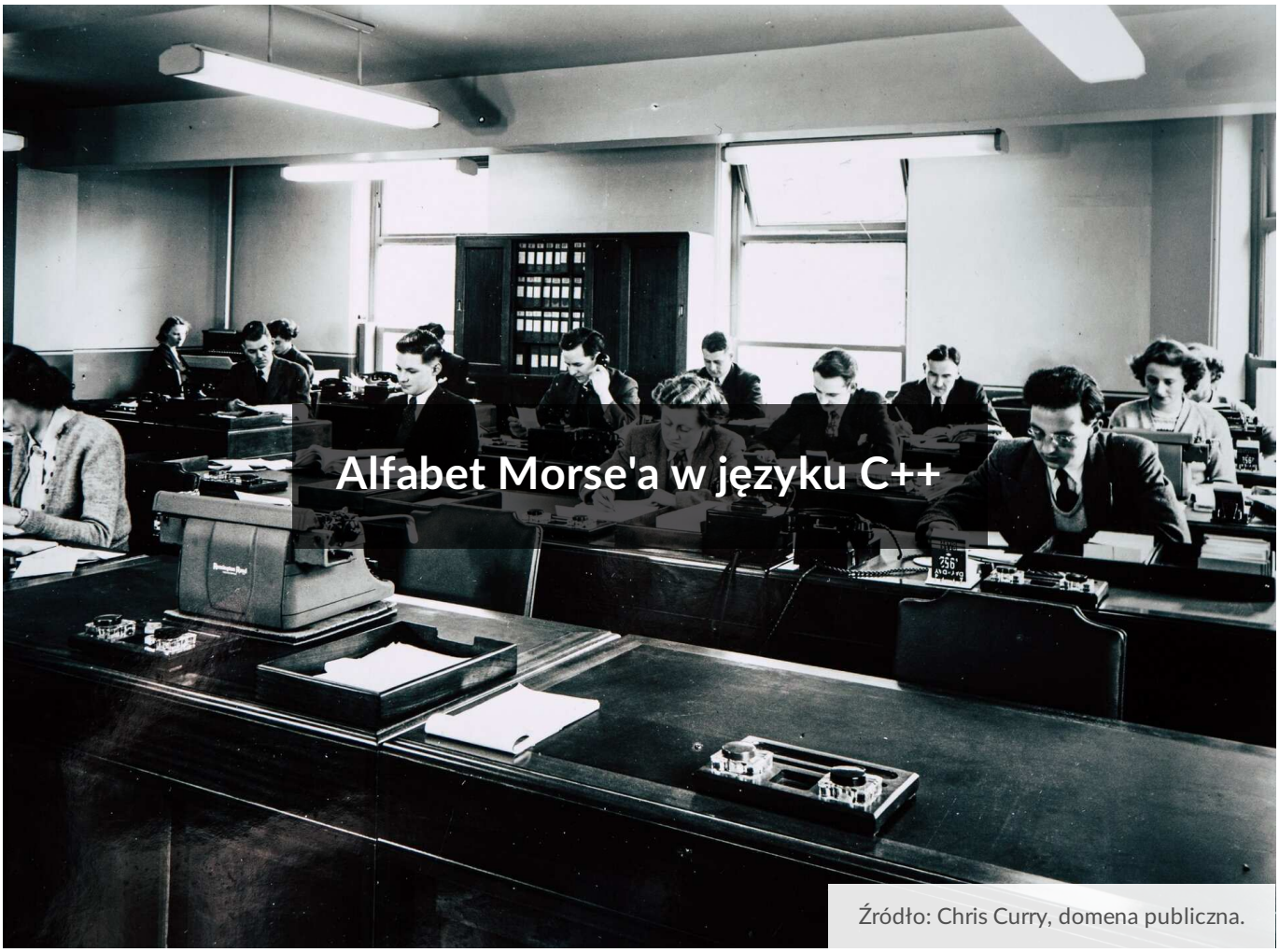
Alfabet Morse'a w języku C++



Alfabet Morse'a w języku C++

Źródło: Chris Curry, domena publiczna.

W e-materiale Alfabet Morse'a przedstawiliśmy najważniejsze informacje dotyczące alfabetu Morse'a. W tym e-materiale zajmiemy się jego implementację w języku C++.

Implementację tego szyfru w pozostałych językach programowania znajdziesz w e-materiałach:

- Alfabet Morse'a w języku Java,
- Alfabet Morse'a w języku Python.

Więcej zadań? Przejdź do e-materiału Alfabet Morse'a – zadania maturalne.

Twoje cele

- Przypomnisz sobie, w jaki sposób koduje się znaki za pomocą alfabetu Morse'a.
- Przeanalizujesz budowę programu napisanego w języku C++, którego zadaniem jest zakodowanie i odkodowanie danego ciągu znaków.
- Wykonasz ćwiczenia wymagające znajomości implementacji kodu Morse'a w języku C++.

Przeczytaj

Alfabet Morse'a – przypomnienie

Zanim przystąpimy do analizy programu dokonującego kodowania zadanego przez użytkownika ciągu znaków na system kodowania alfabetem Morse'a, przypomnijmy, w jaki sposób dokonuje się kodowania.

Tabela kodowań alfabetem Morse'a

Litery, cyfry oraz inne znaki kodowane za pomocą alfabetu Morse'a są zastępowane określonymi ciągami kresek i kropek. Kreska oznacza dłuższy sygnał, kropka krótszy. Poniżej umieszczona została tabela kodowań liter alfabetu łacińskiego, której znajomość okaże się bardzo pomocna przy późniejszej implementacji programów: kodującego i odkodowującego.

Litera alfabetu	Oznaczenie w alfabecie Morse'a	Litera alfabetu	Oznaczenie w alfabecie Morse'a
A	● —	N	— ●
B	— ● ● ●	O	— — —
C	— ● — ●	P	● — — ●
D	— ● ●	Q	— — ● —
E	●	R	● — ●
F	● ● — ●	S	● ● ●
G	— — ●	T	—
H	● ● ● ●	U	● ● —
I	● ●	V	● ● ● —
J	● — — —	W	● — —
K	— ● —	X	— ● ● —
L	● — ● ●	Y	— ● — —
M	— —	Z	— — ● ●

Przykład 1

Spróbujmy zakodować zadany ciąg znaków: „KODOWANIE”. Zwróćmy uwagę na kodowania poszczególnych liter:

Litera alfabetu	Oznaczenie w alfabecie Morse'a	Litera alfabetu	Oznaczenie w alfabecie Morse'a
A	● —	**N**	— ●
B	— ● ● ●	**O**	— — —
C	— ● — ●	P	● — — ●
D	— ● ●	Q	— — ● —
E	●	R	● — ●
F	● ● — ●	S	● ● ●
G	— — ●	T	—
H	● ● ● ●	U	● ● —
I	● ●	V	● ● ● —
J	● — — —	**W**	● — —
K	— ● —	X	— ● ● —
L	● — ● ●	Y	— ● — —
M	— —	Z	— — ● ●

Zakodowany zadany ciąg znaków w alfabecie Morse'a:

— ● — — — — — ● ● — — — ● — — ● — — ● ● ● ●

Implementacja w C++

Przejdźmy teraz do implementacji programu w języku C++.

Kodowanie zadanego ciągu znaków będzie się odbywać w funkcji o nazwie `zakodujCiag()`. Funkcja będzie przyjmować jeden argument – ciąg znaków, który ma zostać poddany kodowaniu. Zarówno zwracana przez funkcję wartość, jak i przyjmowany parametr będą więc łańcuchami znaków `string`.

Ważne!

W przypadku używania w programie łańcuchu znaków `string`, musimy pamiętać o dodaniu odpowiedniej biblioteki:

```
1 #include <string>
```

Następnie, wewnątrz funkcji, zadeklarujmy zmienną, w której będzie przechowywany wynik końcowy kodowania – ciąg kresek i kropek. Od razu dopisujemy instrukcję `return`,

która będzie zwracać wygenerowany napis.

```
std::string zakodujCiag(std::string doZakodowania) {
    std::string zakodowane = "";

    return zakodowane;
}
```

Kolejnym krokiem będzie zadeklarowanie pętli, w której będzie się odbywała iteracja po kolejnych literach. Wewnątrz pętli zadeklarowana została zmienna o nazwie `litera` – będzie ona przechowywać kolejne znaki łańcucha znaków `doZakodowania`.

```
std::string zakodujCiag(std::string doZakodowania) {
    std::string zakodowane = "";

    for (int i = 0; i < doZakodowania.size(); i++) {
        char litera = doZakodowania[i];
    }

    return zakodowane;
}
```

> **Ważne!**
>
> Funkcja wbudowana `size()` zwraca wartość liczbową, równą długości łańcucha znaków.

Za pomocą instrukcji `switch` definiujemy wszystkie kodowania znaków alfabetu Morse'a - dla każdej litery alfabetu łacińskiego przypisany jest odpowiadający ciąg kresek i kropek. Wewnątrz kolejnych bloków instrukcji `case`, dokonywana jest konkatenacja (dołączenie za pomocą znaku +) ciągów kresek i kropek odpowiadających przetwarzanej literze do zmiennej wynikowej `zakodowane`.

```
std::string zakodujCiag(std::string doZakodowania) {
    std::string zakodowane = "";

    for (int i = 0; i < doZakodowania.size(); i++) {
        char litera = doZakodowania[i];

        switch (litera) {
```

```
            case 'A':
                zakodowane += "•-";
                break;
            case 'B':
                zakodowane += "-•••";
                break;
            case 'C':
                zakodowane += "-•-•";
                break;
            case 'D':
                zakodowane += "-••";
                break;
            case 'E':
                zakodowane += "•";
                break;
            case 'F':
                zakodowane += "••-•";
                break;
            case 'G':
                zakodowane += "--•";
                break;
            case 'H':
                zakodowane += "••••";
                break;
            case 'I':
                zakodowane += "••";
                break;
            case 'J':
                zakodowane += "•---";
                break;
            case 'K':
                zakodowane += "-•-";
                break;
            case 'L':
                zakodowane += "•-••";
                break;
            case 'M':
                zakodowane += "--";
                break;
            case 'N':
                zakodowane += "-•";
                break;
```

```
            case 'O':
                zakodowane += "---";
                break;
            case 'P':
                zakodowane += "•--•";
                break;
            case 'Q':
                zakodowane += "--•-";
                break;
            case 'R':
                zakodowane += "•-•";
                break;
            case 'S':
                zakodowane += "•••";
                break;
            case 'T':
                zakodowane += "-";
                break;
            case 'U':
                zakodowane += "••-";
                break;
            case 'V':
                zakodowane += "•••-";
                break;
            case 'W':
                zakodowane += "•--";
                break;
            case 'X':
                zakodowane += "-••-";
                break;
            case 'Y':
                zakodowane += "-•--";
                break;
            case 'Z':
                zakodowane += "--••";
                break;
        }
        zakodowane += " ";
    }

    return zakodowane;
}
```

Po wykonaniu instrukcji `switch` do zmiennej `zakodowane` „doklejany" jest znak spacji. Operacja ta ma na celu oddzielenie od siebie kolejnych ciągów kresek i kropek, czyli kolejnych, zakodowanych, pojedynczych liter.

Kolejnym krokiem jest utworzenie funkcji głównej programu – `main`, w której zostanie wywołana funkcja kodująca oraz zostanie pobrany od użytkownika ciąg znaków do zakodowania.

```
int main() {

    std::string slowo = "";

    std::cout << "Podaj słowo do zakodowania alfabetem Morse'a" <
    std::cin >> slowo;

    std::cout << zakodujCiag(slowo) << std::endl;

    return 0;
}
```

Oto kod całego programu:

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

std::string zakodujCiag(std::string doZakodowania) {
    std::string zakodowane = "";

    for (int i = 0; i < doZakodowania.size(); i++) {
        char litera = doZakodowania[i];

        switch (litera) {
            case 'A':
                zakodowane += "•-";
                break;
            case 'B':
                zakodowane += "-•••";
                break;
            case 'C':
```

```
                zakodowane += "-•-•";
                break;
            case 'D':
                zakodowane += "-••";
                break;
            case 'E':
                zakodowane += "•";
                break;
            case 'F':
                zakodowane += "••-•";
                break;
            case 'G':
                zakodowane += "--•";
                break;
            case 'H':
                zakodowane += "••••";
                break;
            case 'I':
                zakodowane += "••";
                break;
            case 'J':
                zakodowane += "•---";
                break;
            case 'K':
                zakodowane += "-•-";
                break;
            case 'L':
                zakodowane += "•-••";
                break;
            case 'M':
                zakodowane += "--";
                break;
            case 'N':
                zakodowane += "-•";
                break;
            case 'O':
                zakodowane += "---";
                break;
            case 'P':
                zakodowane += "•--•";
                break;
            case 'Q':
```

```
                zakodowane += "--•-";
                break;
            case 'R':
                zakodowane += "•-•";
                break;
            case 'S':
                zakodowane += "•••";
                break;
            case 'T':
                zakodowane += "-";
                break;
            case 'U':
                zakodowane += "••-";
                break;
            case 'V':
                zakodowane += "•••-";
                break;
            case 'W':
                zakodowane += "•--";
                break;
            case 'X':
                zakodowane += "-••-";
                break;
            case 'Y':
                zakodowane += "-•--";
                break;
            case 'Z':
                zakodowane += "--••";
                break;
        }
        zakodowane += " ";
    }

    return zakodowane;
}

int main() {

    std::string slowo = "";

    std::cout << "Podaj słowo do zakodowania alfabetem Morse'a"
    std::cin >> slowo;
```

```
102
103     std::cout << zakodujCiag(slowo) << std::endl;
104
105     return 0;
106 }
```

W kolejnej sekcji poznasz implementację algorytmu odkodowującego zadany przez użytkownika ciąg znaków, zapisany alfabetem Morse'a, na alfabet łaciński.

Słownik

> **alfabet łaciński**
>
> najbardziej popularny na świecie system znaków; składa się on z 26 liter - A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
> **implementacja**
>
> realizacja algorytmu w konkretnym języku programowania

Animacja

Polecenie 1

Zapoznaj się z filmem. Poszukaj informacji na temat znaków interpunkcyjnych, diakrytycznych oraz komend specjalnych w alfabecie Morse'a.

Film dostępny pod adresem /preview/resource/RMEiUBtl2yWNP

Film nawiązujący do treści materiału: Alfabet Morse'a w języku C++.

Kod programu zaprezentowanego w filmie.

Plik o rozmiarze 5.54 KB w języku polskim

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Poniższy program powinien dokonywać poprawnej konwersji ciągu znaków składającego się z liter alfabetu łacińskiego. Brakuje w nim jednak części kodu. Dopisz je w oznaczonych miejscach, aby program działał poprawnie.

Specyfikacja:

Dane:

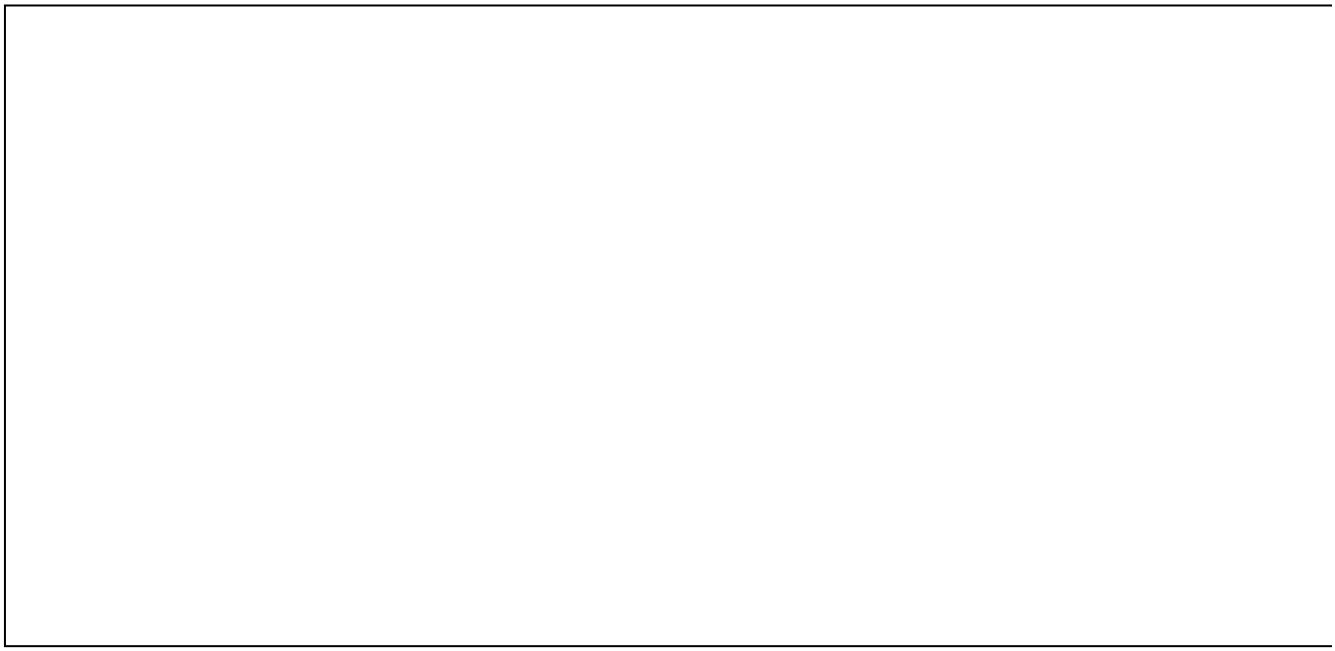
slowo – łańcuch znaków przechowujący słowo do zakodowania kodem Morse'a = „MATURAZINFORMATYKI”

Wynik:

Na standardowym wyjściu program prezentuje zakodowane przy pomocy kodu Morse'a słowo „MATURAZINFORMATYKI”.

Twoje zadania

1. Program powinien zamienić ciąg znaków "MATURAZINFORMATYKI" na alfabet Morse'a.



Ćwiczenie 2

Wykorzystaj kod programu zapisany w Ćwiczeniu 1.

Do kodu programu kodującego łańcuch znaków zapisany w alfabecie łacińskim na alfabet Morse'a dopisz funkcję, która dokona odkodowania z alfabetu Morse'a zakodowanego ciągu znaków. Przetestuj działanie programu dla łańcucha znaków *MATURAZINFORMATYKI*.

Specyfikacja:

Dane:

slowo – łańcuch znaków przechowujący słowo do zakodowania kodem Morse'a

Wynik:

Na standardowym wyjściu program wyświetla w pierwszej linii zakodowane słowo,
a następnie odkodowane.

Przykładowe wyjście:

```
• •-• •-• •- •-• • •••• ••- -- •- -• ••- -- • ••• -
ERRAREHUMANUMEST
```

Twoje zadania

1. Program powinien najpierw wyświetlić zakodowany ciąg znaków z poprzedniego zadania, a następnie za pomocą utworzonej funkcji ma go odkodować i wyświetlić w nowej linii wynik odkodowania.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Alfabet Morse'a w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> IV. Rozwijanie kompetencji społecznych, takich jak: komunikacja i współpraca w grupie, w tym w środowiskach wirtualnych, udział w projektach zespołowych oraz zarządzanie projektami.

Treści nauczania – wymagania szczegółowe

> IV. Rozwijanie kompetencji społecznych.
>
> Zakres podstawowy. Uczeń:
>
> 1) aktywnie uczestniczy w realizacji projektów informatycznych rozwiązujących problemy z różnych dziedzin, przyjmuje przy tym różne role w zespole realizującym projekt i prezentuje efekty wspólnej pracy;

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):

- Przypomnisz sobie, w jaki sposób koduje się znaki za pomocą alfabetu Morse'a.
- Przeanalizujesz budowę programu napisanego w języku C++, którego zadaniem jest zakodowanie i odkodowanie danego ciągu znaków.
- Wykonasz ćwiczenia wymagające znajomości implementacji kodu Morse'a w języku C++.

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;
- 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: „Alfabet Morse'a w języku C++”. Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Przedstawienie tematu zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj” wyświetlone na tablicy.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek”. Uczniowie wspólnie oglądają film, na którym pokazano, w jaki sposób zaimplementować funkcję, która dokona odkodowania zadanego ciągu znaków alfabetu Morse'a na tekst alfabetu łacińskiego.

3. **Ćwiczenie umiejętności.** Prowadzący zapowiada uczniom, że będą rozwiązywać ćwiczenie nr 1 z sekcji „Sprawdź się”. Uczniowie wykonują je w parach. Po ustalonym czasie następuje porównanie napisanych kodów podczas wspólnego omówienia rozwiązań.

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. Na koniec zajęć z programowania w C++ nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak...”.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 2 z sekcji „Sprawdź się”. Na podstawie zdobytej już wiedzy, do kodu programu z ćwiczenia nr 1 tej samej sekcji, mają dopisać funkcję, która dokona odkodowania z alfabetu Morse'a zakodowanego ciągu znaków „MATURAZINFORMATYKI”.

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.