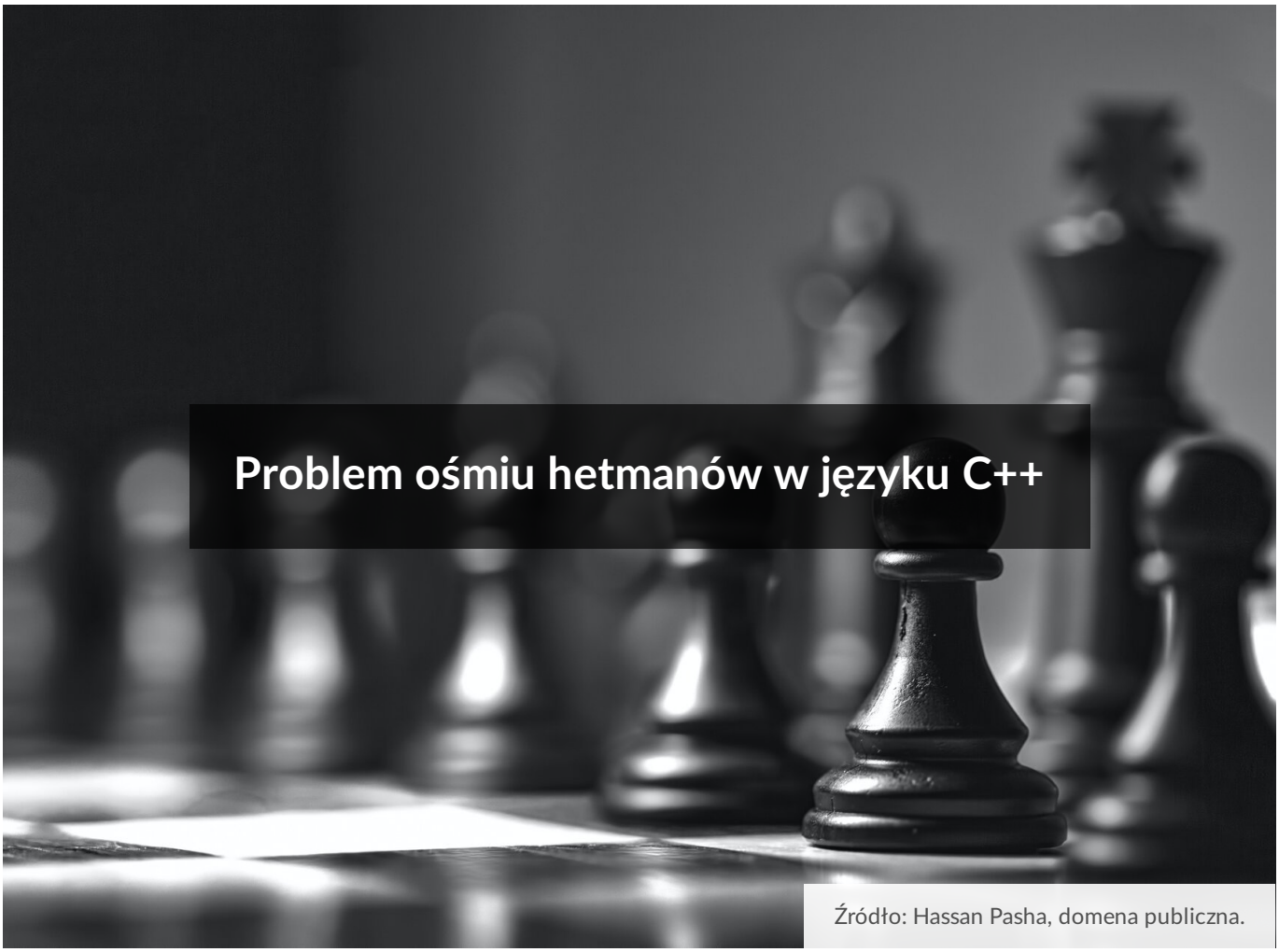
Problem ośmiu hetmanów w języku C++



Problem ośmiu hetmanów w języku C++

Źródło: Hassan Pasha, domena publiczna.

Reguły gry w szachy nie są skomplikowane, jednak każda partia składa się z wielu decyzji taktycznych, co przekłada się na niepowtarzalność rozgrywki. Szachownica i figury szachowe nie muszą jednak służyć wyłącznie do przeprowadzenia klasycznej partii. Istnieje wiele związanych z nimi łamigłówek, których rozwiązanie wymaga ustawienia wybranych figur tak, aby spełnione były konkretne warunki.

Jedno z takich wyzwań znane jest jako problem ośmiu hetmanów. Ten e-materiał poświęcimy napisaniu programu w języku C++ rozwiązującego tę ciekawą zagadkę. Wykorzystamy przy tym rekurencję z nawrotami.

Podstawowe informacje na temat tego zagadnienia zostały przedstawione w e-materiale Problem ośmiu hetmanów. Omówienie implementacji algorytmu w innych językach programowania znajdziesz w pozostałych e-materiałach z tej serii:

- Problem ośmiu hetmanów w języku Java,
- Problem ośmiu hetmanów w języku Python.

Więcej zadań? Zajrzyj do: Problem ośmiu hetmanów – zadania maturalne

> **Twoje cele**
>
> - Wyjaśnisz, na czym polega problem ośmiu hetmanów.

- Przeanalizujesz algorytm rozwiązujący problem ośmiu hetmanów, wykorzystujący nawroty do poprzednich kroków.
- Zaimplementujesz algorytm, odnajdujący rozwiązania problemu ośmiu hetmanów w języku C++ oraz jego uogólnienie dla innych rozmiarów szachownicy.

Przeczytaj

Problem ośmiu hetmanów — notacja

Problem ośmiu hetmanów polega na ustawieniu ośmiu figur szachowych znanych jako hetman lub królowa na klasycznej szachownicy o wymiarach 8x8 w taki sposób, aby nie atakowały się wzajemnie. Hetman to figura, która może poruszać się wzdłuż kolumn, wzdłuż wierszy, a także po skosie, zbijając przy tym inne figury, które stoją na jej drodze. Chcąc ustawić ośmiu hetmanów tak, by nie zbijały się nawzajem, nie można dopuścić do sytuacji, w której dwa z nich znajdą się w jednym wierszu lub w jednej kolumnie. Warunkiem koniecznym (lecz nie wystarczającym) jest więc to, aby każda z ośmiu figur była ustawiona w innej kolumnie – w przeciwnym wypadku przynajmniej dwie figury atakowałyby się wzajemnie. Notacja stosowana do przedstawienia rozwiązania problemu ośmiu hetmanów składa się z ośmiu liczb, z których każda `i`-ta liczba oznacza wiersz, w którym znajduje się hetman z `i`-tej kolumny. Pierwsza liczba stanowi zatem numer wiersza, w którym znajduje się hetman z pierwszej kolumny; druga — numer wiersza, w którym znajduje się hetman z drugiej kolumny, itd.

Implementacja w języku C++

Na początek stwórzmy globalną tablicę typu całkowitego o długości 9. Będzie ona zawierała numery wierszy hetmanów na odpowiednich pozycjach. Komórka o indeksie 0 nie będzie używana.

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

int h[9];
```

Teraz zdefiniujmy funkcję o nazwie `hetman`, przyjmującą parametr będący numerem wiersza, w którym chcemy postawić hetmana. Wewnątrz funkcji stworzymy pętlę, która będzie iterować po wszystkich kolumnach i sprawdzać, czy któraś z nich jest wolna.

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

int h[9];

```

```
void hetman(int wiersz) {
    for (int i = 1; i <= 8; i++) {
        if (h[i] == 0) {
        }
    }
}
```

Jeżeli istnieje wolna kolumna, musimy sprawdzić, czy pole w tym wierszu i w tej kolumnie nie jest atakowane przez innego hetmana po skosie. Tworzymy pętlę, w której będziemy sprawdzać, czy istnieje hetman w innej kolumnie – a jeśli tak, to czy jego zakres ruchu nie zagraża pozycji naszego nowego hetmana. W tym celu używamy wzoru:

$$|a - c| \neq |b - d|$$

gdzie (a, b) to współrzędne pierwszego hetmana, zaś (c, d) to współrzędne drugiego hetmana.

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

int h[9];

void hetman(int wiersz) {
    for (int i = 1; i <= 8; i++) {
        if (h[i] == 0) {
            int czyAtakowane = 0;

            for (int j = 1; j <= 8; j++) {
                if (h[j] && (abs(i - j) == abs(wiersz - h[j]))) {
                    czyAtakowane = 1;
                }
            }
        }
    }
}
```

Jeżeli pole jest atakowane, ustawiamy odpowiednią zmienną na 1. Następnie, jeśli mamy pewność, że pole nie jest zagrożone, umieszczamy hetmana na odpowiedniej pozycji, zapisując numer wiersza w komórce tablicy o indeksie równym numerowi kolumny.

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

int h[9];

void hetman(int wiersz) {
    for (int i = 1; i <= 8; i++) {
        if (h[i] == 0) {
            int czyAtakowane = 0;

            for (int j = 1; j <= 8; j++) {
                if (h[j] && (abs(i - j) == abs(wiersz - h[j]))) {
                    czyAtakowane = 1;
                }
            }

            if (!czyAtakowane) {
                h[i] = wiersz;
            }
        }
    }
}
```

Jeśli nie ustawiliśmy jeszcze wszystkich hetmanów, przechodzimy do kolejnego wiersza, wywołując rekurencyjnie funkcję `hetman(wiersz + 1)`. W momencie zaś, gdy wszystkie osiem wierszy zostanie wypełnione, wypisujemy całe rozwiązanie na standardowe wyjście.

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

int h[9];

void hetman(int wiersz) {
    for (int i = 1; i <= 8; i++) {
        if (h[i] == 0) {
            int czyAtakowane = 0;

            for (int j = 1; j <= 8; j++) {
                if (h[j] && (abs(i - j) == abs(wiersz - h[j]))) {
                    czyAtakowane = 1;
```

```
                }
            }

            if (!czyAtakowane) {
                h[i] = wiersz;

                if (wiersz < 8) {
                    hetman(wiersz + 1);
                } else {
                    for (int i = 1; i <= 8; i++) {
                        cout << h[i];
                    }

                    cout << endl;
                }
            }
        }
    }
}
```

Jeżeli jednak w danym wierszu nie znajdziemy pola, którego nie atakuje inny hetman, musimy powrócić do poprzedniego wiersza i dla kolumny, w której się znajdował, ustawić wartość 0. Dzięki wcześniejszemu zastosowaniu pętli zostanie wyszukane dla niego inne wolne pole. Jeżeli tu także nie znajdziemy innego wolnego pola, nastąpi ponowne cofnięcie itd. Algorytm taki nazywamy **algorytmem z nawrotami**.

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

int h[9];

void hetman(int wiersz) {
    for (int i = 1; i <= 8; i++) {
        if (h[i] == 0) {
            int czyAtakowane = 0;

            for (int j = 1; j <= 8; j++) {
                if (h[j] && (abs(i - j) == abs(wiersz - h[j]))) {
                    czyAtakowane = 1;
                }
            }
```

```
16
17              if (!czyAtakowane) {
18                  h[i] = wiersz;
19
20                  if (wiersz < 8) {
21                      hetman(wiersz + 1);
22                  } else {
23                      for (int i = 1; i <= 8; i++) {
24                          cout << h[i];
25                      }
26
27                      cout << endl;
28                  }
29
30                  h[i] = 0;
31              }
32          }
33      }
34  }
```

Teraz wystarczy w głównej funkcji wypełnić komórki tablicy wartością 0 i wywołać funkcję `hetman` dla pierwszego wiersza. Oto gotowy kod naszego programu, odnajdujący wszystkie możliwe rozwiązania problemu ośmiu hetmanów w języku C++:

```
1  #include <iostream>
2  using namespace std;
3
4  int h[9];
5
6  void hetman(int wiersz) {
7      for (int i = 1; i <= 8; i++) {
8          if (h[i] == 0) {
9              int czyAtakowane = 0;
10
11             for (int j = 1; j <= 8; j++) {
12                 if (h[j] && (abs(i - j) == abs(wiersz - h[j]))) {
13                     czyAtakowane = 1;
14                 }
15             }
16
17             if (!czyAtakowane) {
```

```
                h[i] = wiersz;

                if (wiersz < 8) {
                    hetman(wiersz + 1);
                } else {
                    for (int i = 1; i <= 8; i++) {
                        cout << h[i];
                    }

                    cout << endl;
                }

                h[i] = 0;
            }
        }
    }
}

int main() {
    for (int i = 1; i <= 8; i++) {
        h[i] = 0;
    }

    hetman(1);
    return 0;
}
```

Problem n hetmanów

Problem n hetmanów to termin, którym określamy zastosowanie problemu ośmiu hetmanów dla szachownicy mniejszej lub większej niż standardowa (8x8 pól). W problemie n hetmanów chcemy ustawić n figur na szachownicy o wymiarach nxn w taki sposób, aby nie atakowały się wzajemnie.

Poniższa tabela prezentuje liczbę rozwiązań problemu n hetmanów dla danych wartości n:

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
liczba rozwiązań	1	0	0	2	10	4	40	92	352	724	2680	14200

Praca domowa

> Korzystając z zaprezentowanego programu, dokonaj jego uogólnienia i napisz program, który rozwiąże problem dla n hetmanów.

Słownik

> **warunek konieczny**
>
> warunek (zazwyczaj jeden z wielu), którego spełnienie jest konieczne dla zaistnienia danego faktu
> **warunek wystarczający**
>
> warunek, którego spełnienie wystarcza dla zaistnienia danego faktu
> **rekurencja**
>
> technika programowania, w której funkcja odwołuje się do samej siebie
> **algorytm z nawrotami**
>
> algorytm, w którym zapisujemy kolejne kroki rozwiązania; gdy okaże się, że dany krok nie prowadzi do prawidłowego rozwiązania całego problemu, następuje powrót do poprzedniego etapu i szukanie innego wyjścia

Symulacja interaktywna

Polecenie 1

Przeanalizuj symulację interaktywną, prezentującą kolejne kroki rozwiązania problemu ośmiu hetmanów dla sytuacji, w której trzech hetmanów znajduje się już w konfiguracji {10003020}.

Następnie spróbuj na kartce wykonać samodzielnie to zadanie dla sytuacji, w której czterej hetmani zostali już rozmieszczeni w następujący sposób: {03100024}.

Polecenie 2

Przygotuj notatkę ze swoimi spostrzeżeniami dotyczącymi rozwiązania przedstawionego w symulacji.

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program rozwiązujący problem ośmiu hetmanów i wyświetlający liczbę wszystkich możliwych ustawień hetmanów dla planszy wielkości `nxn`. Działanie programu przetestuj dla planszy o wymiarach 8×8.

Specyfikacja problemu:

Dane:

- n – liczba naturalna

Wynik:

- `licznik` – liczba naturalna będąca liczbą ustawień n hetmanów na planszy `nxn`

Twoje zadania

1. Program na standardowym wyjściu drukuje liczbę rozwiązań problemu ośmiu hetmanów.

Ćwiczenie 2

Napisz program wyświetlający ustawienie nieatakujących się n hetmanów dla sytuacji, w której dwaj hetmani mają już przydzielone pozycje.
Przetestuj program dla ośmiu hetmanów, z których dwaj mają następujące współrzędne:
`(x1,y1) = (1,8)`
`(x2,y2) = (8,5)`

Specyfikacja problemu:

Dane:

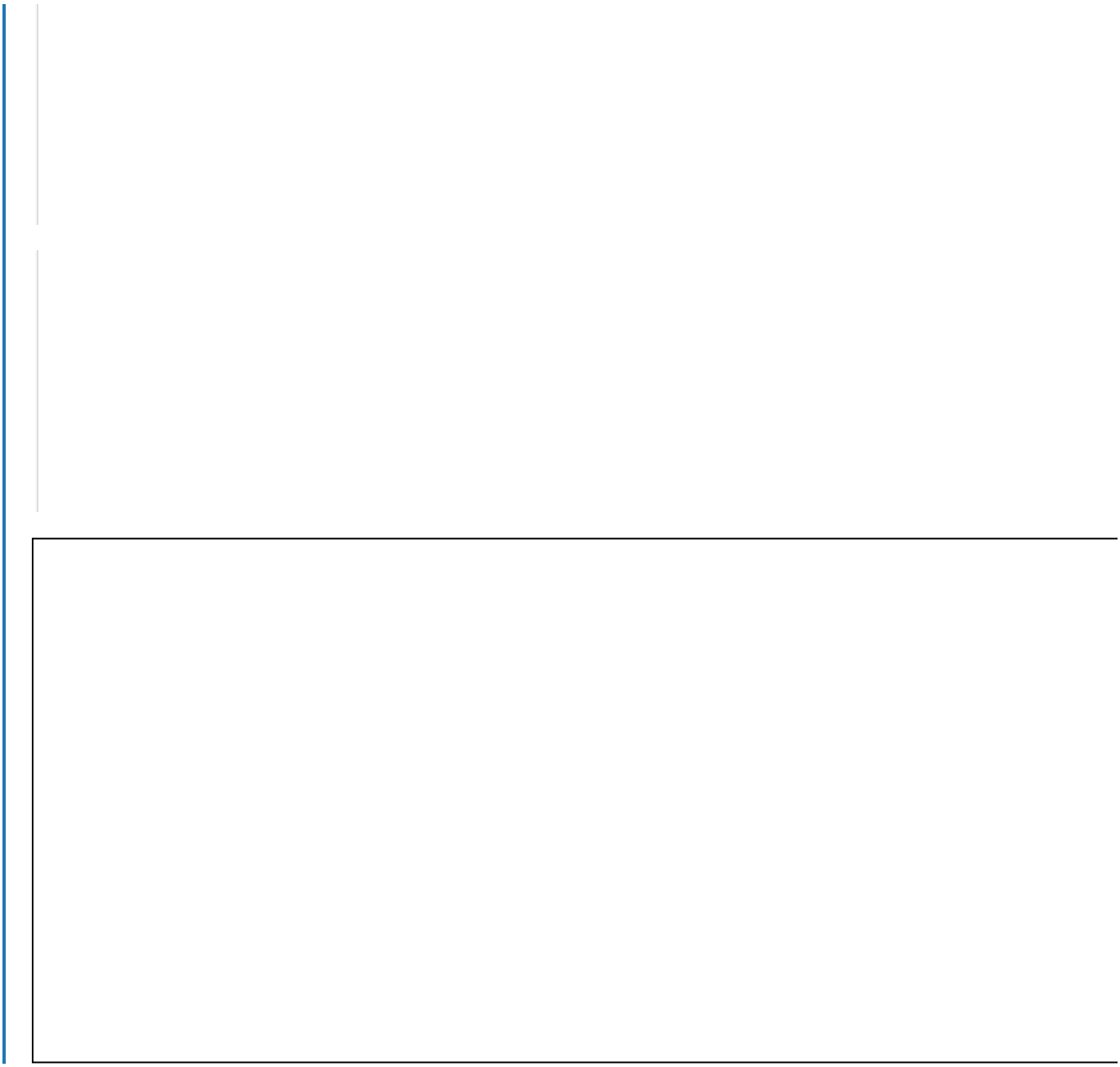
- `n` – liczba naturalna
- `(x1, y1)` – para liczb naturalnych dodatnich
- `(x2, y2)` – para liczb naturalnych dodatnich

Wynik:

Na standardowym wyjściu wyświetlany jest ciąg liczb naturalnych będący kodem znalezionego ustawienia hetmanów; `i`-ta liczba oznacza numer wiersza, w którym znajduje się hetman w `i`-tej kolumnie.

Twoje zadania

1. Na standardowym wyjściu wyświetlany jest ciąg liczb naturalnych będący kodem znalezionego ustawienia hetmanów; `i`-ta liczba oznacza numer wiersza, w którym znajduje się hetman w `i`-tej kolumnie.



Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Problem ośmiu hetmanów w języku C++

Grupa docelowa:

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

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);

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):

- Wyjaśnisz, na czym polega problem ośmiu hetmanów.
- Przeanalizujesz algorytm rozwiązujący problem ośmiu hetmanów, wykorzystujący nawroty do poprzednich kroków.
- Zaimplementujesz algorytm, odnajdujący rozwiązania problemu ośmiu hetmanów w języku C++ oraz jego uogólnienie dla innych rozmiarów szachownicy.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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: „Problem ośmiu hetmanów w języku C++”. Uczniowie zapoznają się z treściami w sekcji „Przeczytaj”.

Faza wstępna:

1. Nauczyciel wyświetla temat oraz cele zajęć, omawiając lub ustalając razem z uczniami 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.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Przeczytaj". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel czyta polecenie nr 1 z sekcji „Symulacja interaktywna". Prosi uczniów, aby wykonali je w parach. Następnie wybrana osoba prezentuje propozycję odpowiedzi, a pozostali uczniowie ustosunkowują się do niej. Nauczyciel w razie potrzeby uzupełnia ją, udziela też uczniom informacji zwrotnej.
3. W kolejnym etapie uczniowie dobierają się w pary i wykonują ćwiczenia nr 1 i 2 z sekcji „Sprawdź się". Następnie konsultują swoje rozwiązania z inną parą uczniów i ustalają jedną wersję odpowiedzi.

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. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku C++.

Praca domowa:

1. Poszukaj informacji na temat złożoności czasowej oraz obliczeniowej poznanego algorytmu. Ile czasu potrzebowałby współczesny komputer, żeby rozwiązać problem ośmiu hetmanów? Ile komputer, który wysłał ludzi w kosmos?

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 treści w sekcjach: „Przeczytaj", „Symulacja interaktywna", „Sprawdź się" jako materiał do lekcji powtórkowej.