Algorytmy tekstowe

Algorytmy tekstowe mają decydujące znaczenie przy wyszukiwaniu informacji typu tekstowego, ten typ informacji jest szczególnie popularny w informatyce, np. w edytorach tekstowych i wyszukiwarkach internetowych. Tekst jest ciągiem symboli.

Przyjmujemy, że jest on zadany tablicą $x[1, \ldots, n]$, elementami której są symbole ze skończonego zbioru *A* (zwanego alfabetem).

Liczba $n = |x|$ jest długością (rozmiarem) tekstu. W większości naszych algorytmów jedyne operacje dopuszczalne na symbolach wejściowych to porównania dwóch symboli.

Wstęp do algorytmu Knutha-Morrisa-Pratta

https://zpe.gov.pl/a/przeczytaj/DcT6FBT2u

Gra w wisielca w C++

Wiemy, jak przebiega partia gry w wisielca, spróbujmy więc napisać program, który ją odzwierciedli. Przedstawimy alternatywę dla realizacji kodu gry, zaprezentowanej w filmie samouczku.

Pierwszym krokiem jest zadeklarowanie dwóch zmiennych:

- int zycia = 10; - będzie przechowywać pozostałą liczbę prób (pozostałe życia), służącą do odgadnięcia litery lub całego słowa,
- string nadpisane = ""; - będzie przechowywać słowo z aktualnie odgadniętymi literami.

Kolejnym krokiem jest pobranie od pierwszego gracza słowa, które gracz drugi będzie próbował odgadnąć (w kodzie zaprezentowanym w filmie słowo było losowane z predefiniowanej tablicy haseł). Ciąg znaków pobrany z klawiatury zapisujemy w zmiennej slowo.

```
1 string slowo;
2 cin >> slowo;
```

Następnie wypełniamy zmienną nadpisane znakami. Liczba znaków podkreślenia odpowiada długości słowa wymyślonego przez gracza pierwszego. Zmienną wypisujemy na ekran, aby zobaczył ją gracz drugi i wiedział jaka jest długość słowa. Zadeklarowana zostaje także zmienna logiczna czyOdgadniete. Od jej wartości zależeć będzie, czy gracz drugi odgadł już słowo czy nie.

```
1 for (int i = 0; i < slowo.length(); i++) {
2     nadpisane += "_";
3 }
4
5 cout << nadpisane << endl;
6
7 bool czyOdgadniete = false;
8 string odgadywanie;
```

Teraz należy zadeklarować pętlę, w której będzie się odbywała najważniejsza część algorytmu – pobieranie od gracza drugiego kolejnych liter (lub całego słowa), które następnie zostaną sprawdzone, czy są częścią wymyślonego słowa.

```
while (zycia > 0) {
    cout << "Podaj literę lub całe słowo do sprawdzenia" << endl;
    cin >> odgadywanie;
```

Ponieważ gracz może zgadywać pojedynczą literę lub całe słowo (wersja gry opisywana w filmie zakładała odgadywanie jedynie pojedynczych liter), to należy obsłużyć oba przypadki za pomocą instrukcji warunkowej if ... else.

```
if (odgadywanie.length() == 1) {

} else if(odgadywanie.length() > 1) {

}
```

Zajmijmy się teraz pierwszym przypadkiem, tzn. użytkownik odgaduje pojedynczą literę. Wewnątrz instrukcji warunkowej sprawdzane jest, czy podana przez gracza litera występuje w wymyślonym słowie. Jeżeli występuje, zmienna czyZawiera przyjmuje wartość true i następuje natychmiastowe wyjście z pętli, natomiast w przeciwnym przypadku przyjmuje wartość false.

```
bool czyZawiera = false;

if (odgadywanie.length() == 1) {
    for (int i = 0; i < slowo.length(); i++) {
        if (slowo[i] == odgadywanie[0]) {
            czyZawiera = true;
            break;
        }
    }

} else if(odgadywanie.length() > 1) {

}
```

Kolejnym krokiem jest sprawdzenie wartości zmiennej czyZawiera i zdefiniowanie odpowiednich operacji:

- Jeżeli zapisana w zmiennej wartość logiczna to true:

- nadpisywana jest zmienna nadpisane – znaki podkreślenia w odpowiednich miejscach zamieniane są na rzeczywiste odgadnięte litery.

- Jeżeli zapisana w zmiennej wartość logiczna to false:

- graczowi odjęte zostaje jedno życie i wyświetla się komunikat informujący o tym, że litera którą podał, nie występuje w słowie wymyślonym przez gracza pierwszego.

```
bool czyZawiera = false;

if (odgadywanie.length() == 1) {
    for (int i = 0; i < slowo.length(); i++) {
        if (slowo[i] == odgadywanie[0]) {
            czyZawiera = true;
            break;
        }
    }

    if (czyZawiera == true) {
        for (int i = 0; i < slowo.length(); i++) {
            if (slowo[i] == odgadywanie[0]) {
                nadpisane[i] = odgadywanie[0];
            }
        }
    } else {
        zycia--;
        cout << "Słowo nie zawiera litery " + odgadywanie << endl;
    }

} else if(odgadywanie.length() > 1) {

}
```

Następnie sprawdzane jest, czy słowo zostało w pełni odgadnięte, czy też zawiera jeszcze znaki podkreślenia. Od zmiennej czyOdgadniete zależy to, czy wyjdziemy z pętli i zakończymy algorytm.

```cpp
bool czyZawiera = false;

if (odgadywanie.length() == 1) {
    for (int i = 0; i < slowo.length(); i++) {
        if (slowo[i] == odgadywanie[0]) {
            czyZawiera = true;
            break;
        }
    }

    if (czyZawiera == true) {
        for (int i = 0; i < slowo.length(); i++) {
            if (slowo[i] == odgadywanie[0]) {
                nadpisane[i] = odgadywanie[0];
            }
        }
    } else {
        zycia--;
        cout << "Słowo nie zawiera litery " + odgadywanie << endl;
    }

    czyOdgadniete = true;

    for (int i = 0; i < nadpisane.length(); i++) {
        if (nadpisane[i] == '_') {
            czyOdgadniete = false;
            break;
        }
    }

    if (czyOdgadniete == true) {
        break;
    }

} else if(odgadywanie.length() > 1) {

}
```

Teraz zdefiniujmy operacje, których wykonywanie nastąpi, gdy użytkownik nie wprowadzi pojedynczej litery tylko ciąg znaków.

Sprawdzane jest, czy ciąg znaków podany przez gracza jest taki sam jak ciąg znaków wymyślony przez gracza pierwszego, tj. czy użytkownik poprawnie odgadł całe słowo. W przypadku, gdy wyrazy te są identyczne, oznacza to, że gracz drugi poprawnie odgadł słowo i następuje wyjście z pętli.
W przeciwnym wypadku dekrementujemy zmienną zycia, która określa liczbę pozostałych prób.

```cpp
bool czyZawiera = false;

if (odgadywanie.length() == 1) {
    for (int i = 0; i < slowo.length(); i++) {
        if (slowo[i] == odgadywanie[0]) {
            czyZawiera = true;
            break;
        }
    }

    if (czyZawiera == true) {
        for (int i = 0; i < slowo.length(); i++) {
            if (slowo[i] == odgadywanie[0]) {
                nadpisane[i] = odgadywanie[0];
            }
        }
    } else {
        zycia--;
        cout << "Słowo nie zawiera litery " + odgadywanie << endl;
    }
```

```
21
22      czyOdgadniete = true;
23
24      for (int i = 0; i < nadpisane.length(); i++) {
25          if (nadpisane[i] == '_') {
26              czyOdgadniete = false;
27              break;
28          }
29      }
30
31      if (czyOdgadniete == true) {
32          break;
33      }
34
35  } else if(odgadywanie.length() > 1) {
36
37      if (odgadywanie == slowo) {
38          czyOdgadniete = true;
39          break;
40      } else {
41          zycia--;
42          cout << "Podane słowo nie jest odgadywanym hasłem" << endl;
43      }
44  }
```

Po każdej iteracji pętli wyświetlane są dwie informacje:

- zmienna nadpisane, czyli słowo wypełnione o odgadnięte litery,
- informacja o pozostałych życiach (próbach).

```
1 cout << nadpisane << endl;
2 cout << "Pozostałe życia: " + to_string(zycia) << endl;
```

Ostatnie, co należy zrobić (już poza główną pętlą), to wyświetlić komunikat o przebiegu gry:

```
1 if (czyOdgadniete == false) {
2     cout << "Niestety nie udało ci się odgadnąć słowa" << endl;
3 } else {
4     cout << "Udało ci się odgadnąć słowo!" << endl;
5 }
```