АЛГОРИТМЫ ПОИСКА В ТЕКСТЕ

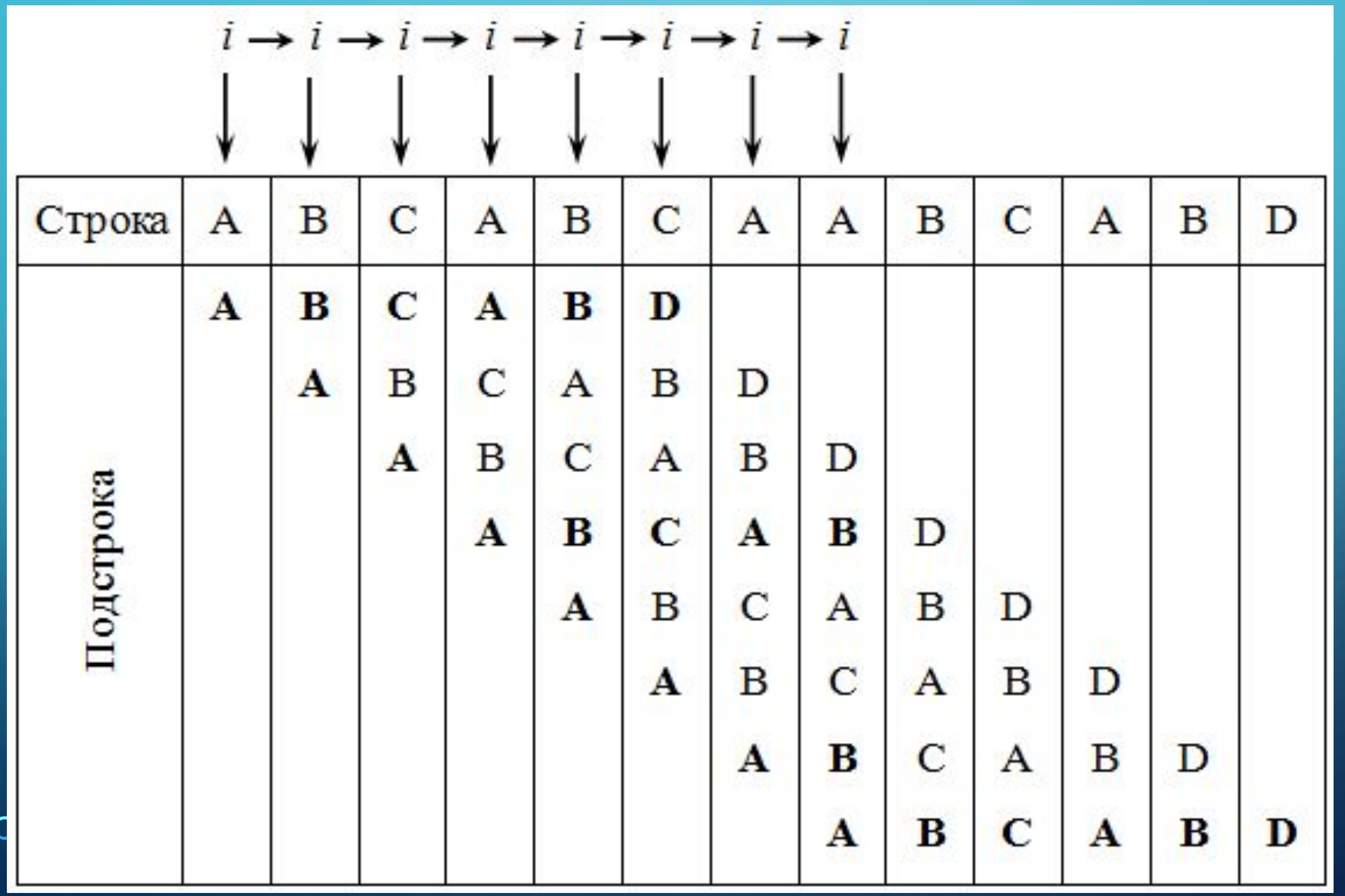
- Строку будем обозначать символами алфавита, например X=x[1]x[2]...x[n] – строка длиной n, где x[i] – i -ый символ строки X, принадлежащий алфавиту. Строка, не содержащая ни одного символа, называется *пустой*.
- Строка X называется **подстрокой** строки Y, если найдутся такие строки Z_1 и Z_2, что $Y=Z_1XZ_2$. При этом Z_1 называют *левым*, а Z_2 – *правым крылом подстроки*. Подстрокой может быть и сама строка. Иногда при этом строку X называют вхождением в строку Y. Например, строки hrf и fhr является подстроками строки abhrfhr.

- Подстрока X называется **префиксом** строки Y, если есть такая подстрока Z, что Y=XZ. Причем сама строка является префиксом для себя самой (так как найдется нулевая строка L, что X=XL). Например, подстрока ab является префиксом строки abcfa.

- Подстрока X называется **суффиксом** строки Y, если есть такая подстрока Z, что Y=ZX. Аналогично, строка является суффиксом себя самой. Например, подстрока bfg является суффиксом строки vsenfbfg.

ПРЯМОЙ ПОИСК

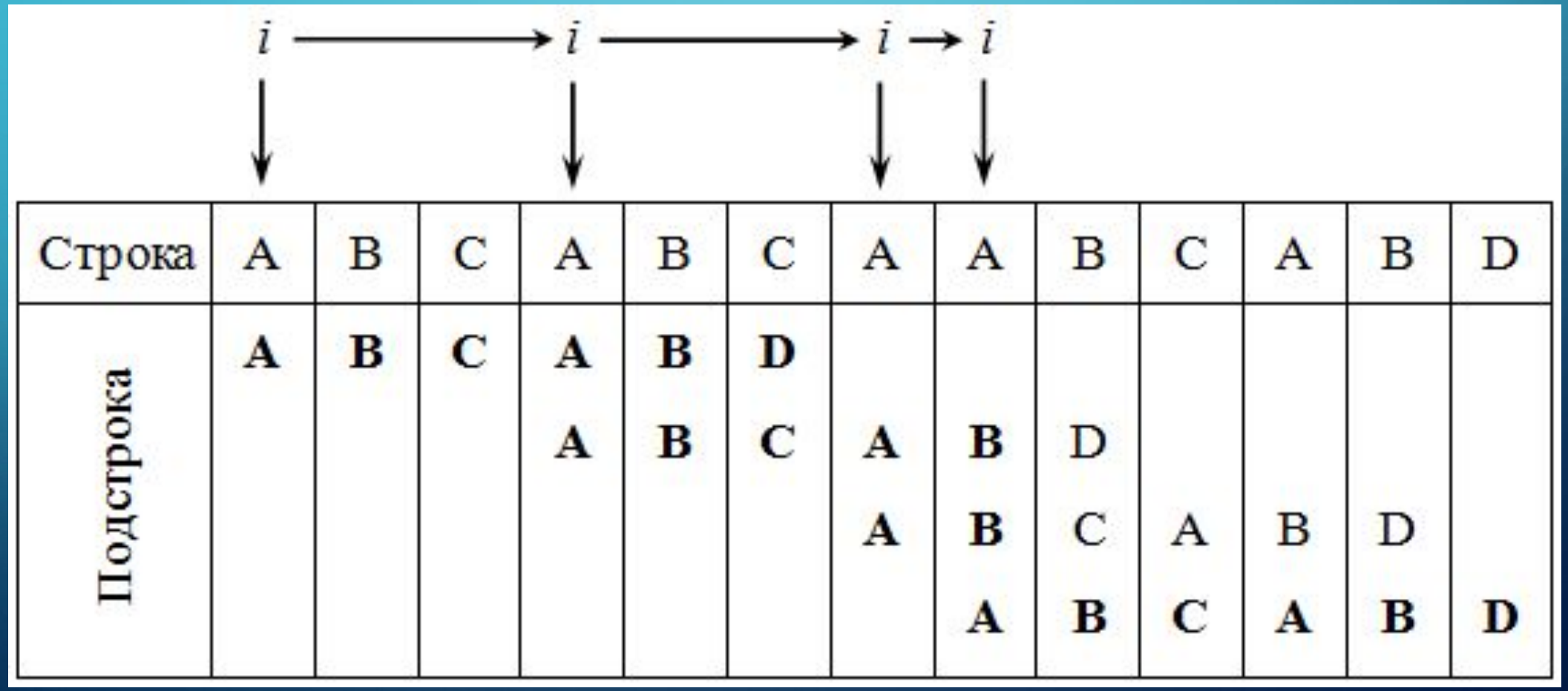
O((N-M+1)XM)



```
//Описание функции прямого поиска подстроки в строке
int DirectSearch(char *string, char *substring){
  int sl, ssl;
  int res = -1;
  sl = strlen(string);
  ssl = strlen(substring);
  if ( sl == 0 )
    cout << "Неверно задана строка\n";
  else if ( ssl == 0 )
    cout << "Неверно задана подстрока\n";
  else
    for (int i = 0; i < sl - ssl + 1; i++)
      for (int j = 0; j < ssl; j++)
```

АЛГОРИТМ КНУТА, МОРРИСА И ПРАТТА O(M+N)

- они совместно опубликовали в 1977 году. Алгоритм Кнута, Морриса и Пратта (КМП-алгоритм) требует только O(n) сравнений даже в самом худшем случае.

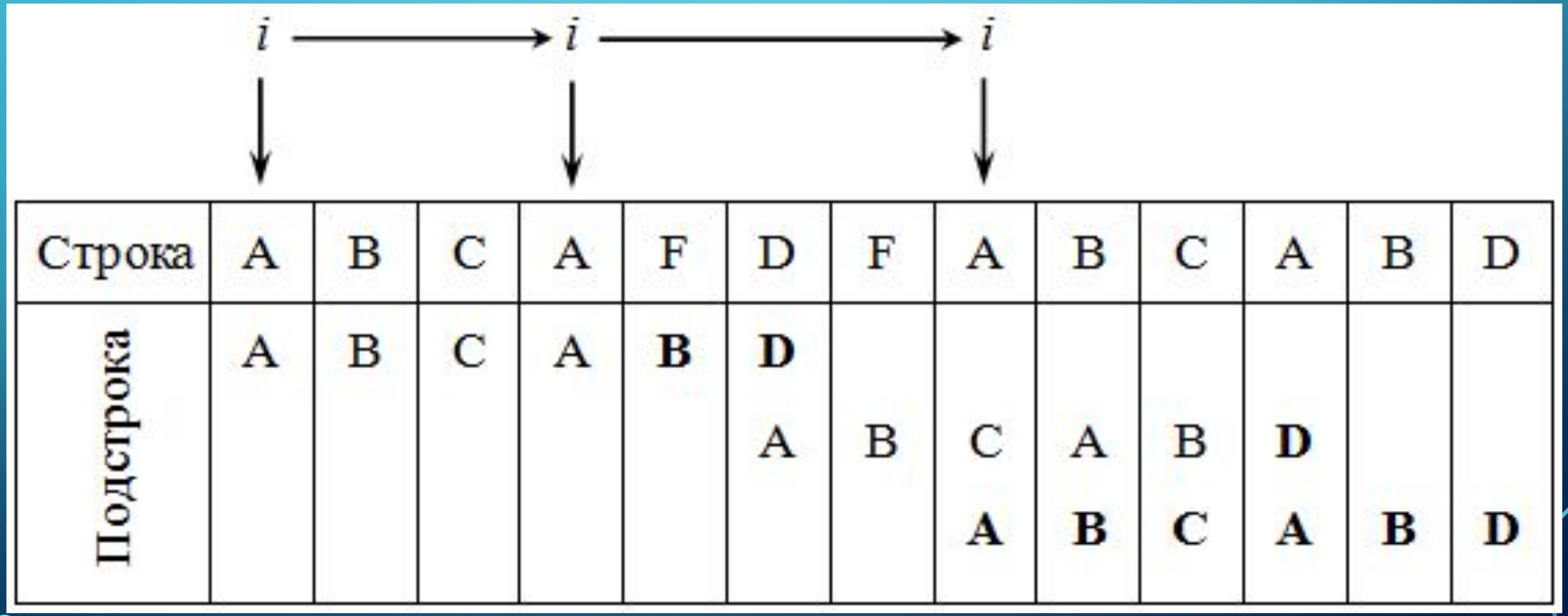


Строка	A	B	C	A	B	C	A	A	B	C	A	B	D
Подстрока	**A**	**B**	**C**	**A** **A**	**B** **B**	**D** **C**	**A** **A**	**B** **B** **A**	**D** **C** **B**	**A** **C**	**B** **A**	**D** **B**	**D**

```
//описание функции алгоритма Кнута, Морриса и Пратта
int KMPSearch(char *string, char *substring){
  int  sl, ssl;
  int res = -1;
  sl = strlen(string);
  ssl = strlen(substring);
  if ( sl == 0 )
    cout << "Неверно задана строка\n";
  else if ( ssl == 0 )
    cout << "Неверно задана подстрока\n";
  else {
    int  i, j = 0, k = -1;
    int  *d;
    d = new int[1000];
    d[0] = -1;
    while ( j < ssl - 1 ) {
```

АЛГОРИТМ БОЙЕРА И МУРА

- разработан Р. Бойером и Д. Муром в 1977 году
- наихудший случай O(m+n)



Строка	A	B	C	A	F	D	F	A	B	C	A	B	D
Подстрока	A	B	C	A	**B**	**D** A	B	C **A**	A **B**	B **C**	**D** **A**	**B**	**D**

```
//описание функции алгоритма Бойера и Мура
int BMSearch(char *string, char *substring){
  int  sl, ssl;
  int res = -1;
  sl = strlen(string);
  ssl = strlen(substring);
  if ( sl == 0 )
    cout << "Неверно задана строка\n";
  else if ( ssl == 0 )
    cout << "Неверно задана подстрока\n";
  else {
    int  i, Pos;
    int  BMT[256];
    for ( i = 0; i < 256; i ++ )
      BMT[i] = ssl;
    for ( i = ssl-1; i >= 0; i-- )
      if ( BMT[(short)(substring[i])] == ssl )
        BMT[(short)(substring[i])] = ssl - i - 1;
    Pos = ssl - 1;
```