Data Structures And Applications in C

MODULE 1

- **INTRODUCTION TO DATA STRUCTURES:** Data Structures, Classifications (Primitive & Non-Primitive), Data structure Operations
- **Review of** pointers and dynamic Memory Allocation,
- **ARRAYS and STRUCTURES:** Arrays, Dynamic Allocated Arrays, Structures and Unions, Polynomials, Sparse Matrices, representation of Multidimensional Arrays, Strings.
- **STACKS:** Stacks, Stacks Using Dynamic Arrays, Evaluation and conversion of Expressions.

PATTERN MATCHING

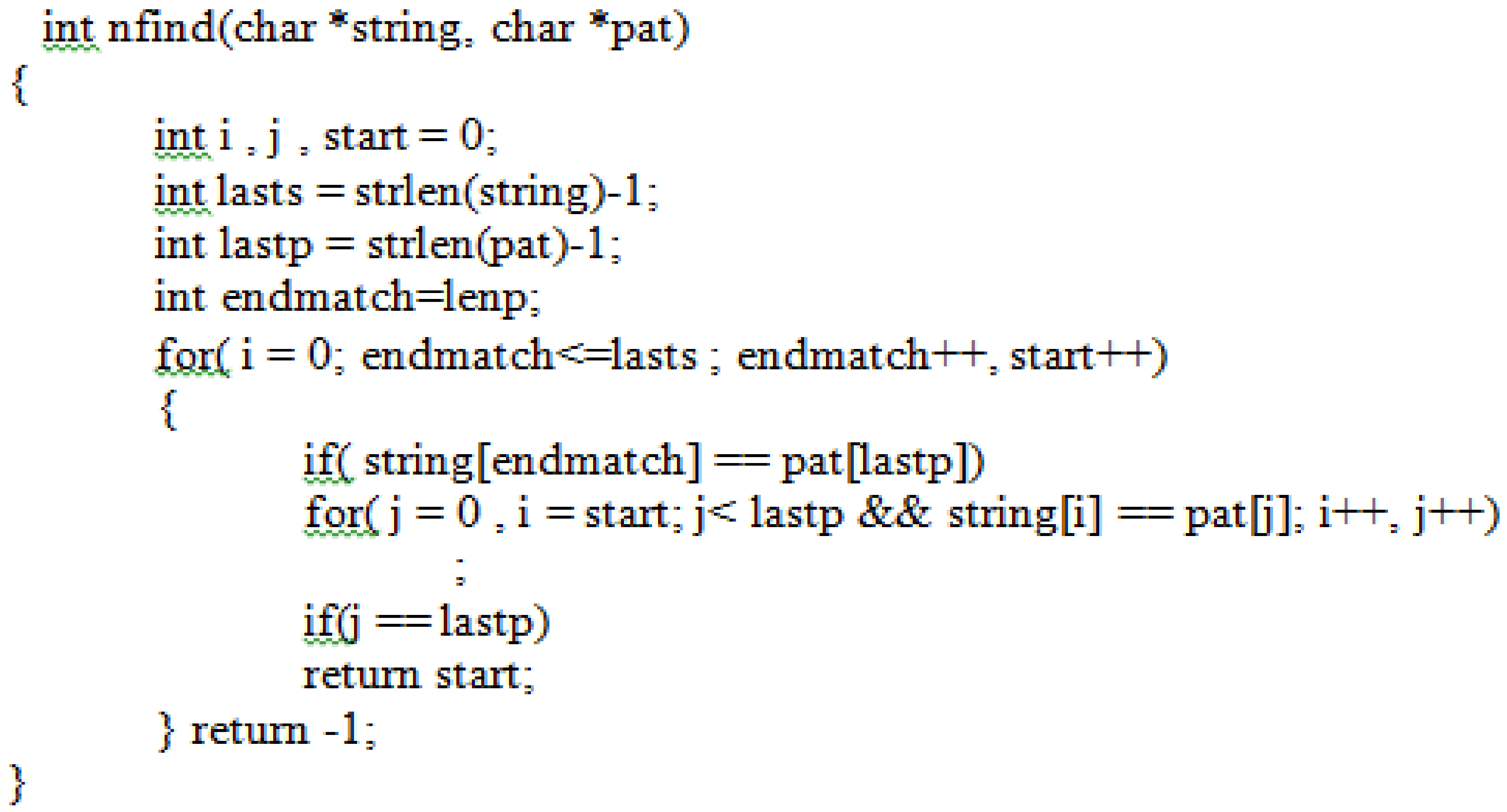
- C programming code to check if a given string is present in another string
- For example the string "programming" is present in "c programming"
- If the string is present then it's location (i.e. at which position it is present) is printed
- We create a function match which receives two character pointers and return the position if matching occurs otherwise returns -1
- naive string search algorithm is implemented in this c program

BRUTE-FORCE ALGORITHM (NATIVE/NAÏVE STRING SEARCH ALGORITHM)

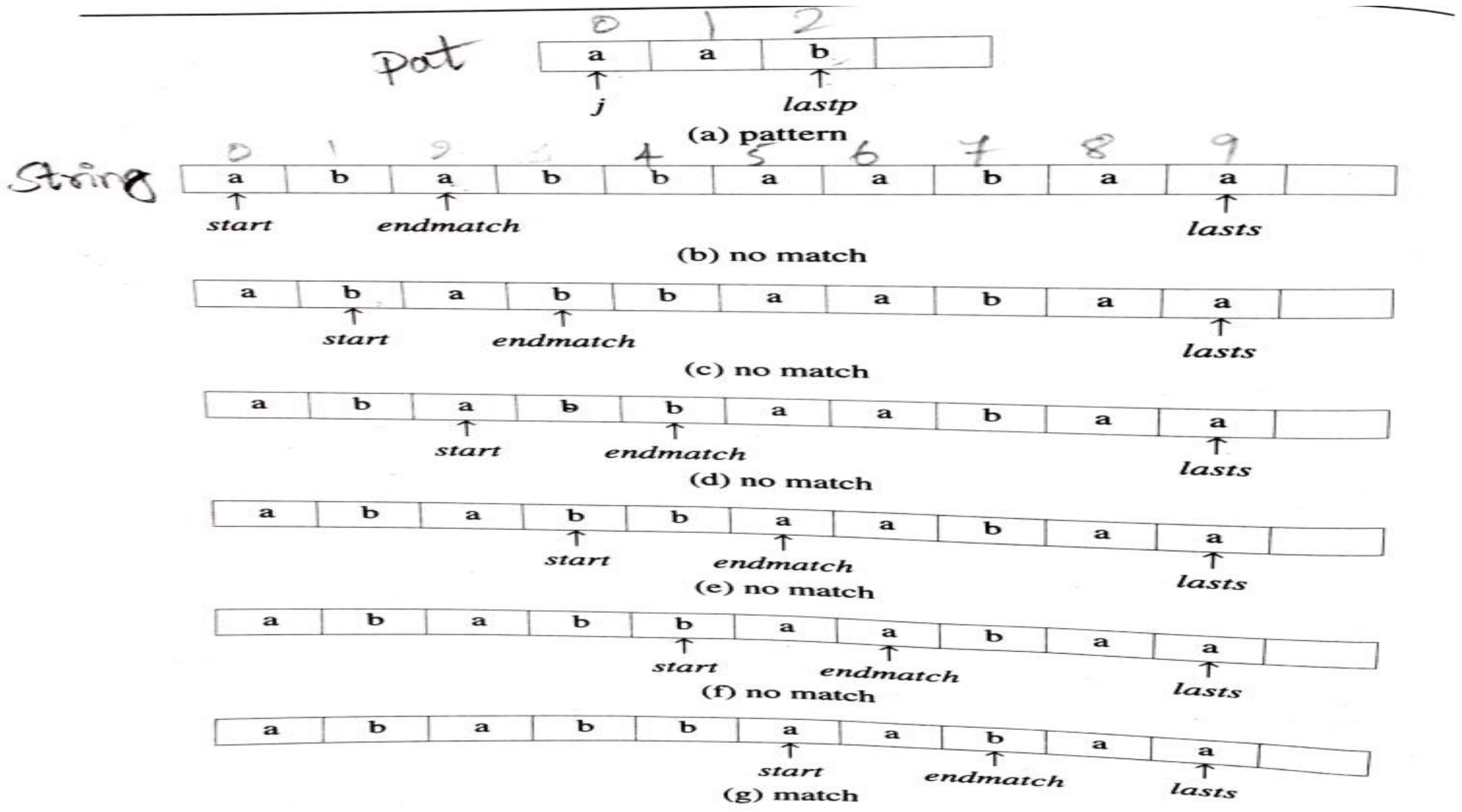
```
Algorithm_Brute_force_pattern_matching(T[0…..n-1], P[0…..m-1])
{
        for(i=0; i<=n-m; i++)
        {
                j=0;
                while(j<m && T[i+j]==P[j])
                        j=j+1;
                if(j==m)
                        return i;
        }
        return -1;
}
```

- https://youtu.be/fCJBtVG2kQo?si=DHW7UtYnsXajaBtG
- https://www.youtube.com/watch?v=fCJBtVG2kQo

PATTERN MATCHING BY CHECKING END INDICES FIRST



```
   int nfind(char *string, char *pat)
 {
        int i , j , start = 0;
        int lasts = strlen(string)-1;
        int lastp = strlen(pat)-1;
        int endmatch=lenp;
        for( i = 0; endmatch<=lasts ; endmatch++, start++)
        {
                if( string[endmatch] == pat[lastp])
                for( j = 0 , i = start; j< lastp && string[i] == pat[j]; i++, j++)
                        ;
                if(j == lastp)
                return start;
        } return -1;
 }
```


Pat
0 1 2
a a b
j
lastp
(a) pattern
String
0 1 2 3 4 5 6 7 8 9
a b a b b a a b a a
start
endmatch
lasts
(b) no match
start
endmatch
lasts
(c) no match
start
endmatch
lasts
(d) no match
start
endmatch
lasts
(e) no match
start
endmatch
lasts
(f) no match
start
endmatch
lasts
(g) match

KNUTH MORRIS PRATT (KMP) PATTERN MATCHING ALGORITHM

```c
#include <stdio.h>
#include <string.h>
#define max_string_size 100
#define max_pattern_size 100
int pmatch();
void fail();
int failure[max_pattern_size];
char string[max_string_size];
char pat[max_pattern_size];
```

```c
int pmatch(char *string, char *pat)
{
/* Knuth, Morris, Pratt string matching algorithm */
  int i = 0, j = 0;
  int lens = strlen(string);
  int lenp = strlen(pat);
  while ( i < lens && j < lenp ) {
    if (string[i] == pat[j]) {
      i++; j++; }
    else if (j == 0) i++;
         else j = failure[j-1]+1;
  }
  return ( (j == lenp) ? (i-lenp) : -1);
}
```

- https://www.youtube.com/watch?v=pu2aO_3R118

OR
KNUTH MORRIS PRATT (KMP) PATTERN MATCHING ALGORITHMS

```
Algorithm_pattern_matching_KMP
{
        k=1, $S_1=Q_0$, n=length(s);
        While(k<=n && $S_k \neq P$)
        {
                Read $t_k$ ;
                $S_{k+1}$ =F($S_k$, $t_k$) ;
                k = k+1;
        }
        if(k>n)
                index = 0;
        else
                index = k – length(p);
        return index;
}
```

END OF INTRODUCTION TO DATA STRUCTURES