|  |
| --- |
| #include <stdio.h>  #define FOR(i, x, y) for(int i = x;i <=y; i++)  #define RE(i, x, y) for(int i = y; i>= x;i --)  #define maxN 100  int arr\_coppy(int \*arr, int \*arr1, int arr\_size);  int main()  {  int n = 4;  int a[maxN] = {1,2,2,3};  int b[maxN] = {};  arr\_coppy(a, b, n);  FOR(i, 0, n - 1)  {  printf("%d ", b[i]);  }  return 0;  }  /\*  ----------COPPY ARRAY---------------  Paramaters:  - \*arr1 : the source array  - \*arr2 : the destination array  - arr\_size : size of the array  Function:  - Used for copying data from the source array to the destination array.  It copies elements with indexes from 0 to arr\_size - 1. The ouput is 0 if the function not error.  \*/  int arr\_coppy(int \*arr1, int \*arr2, int arr\_size)  {  FOR(i, 0, arr\_size - 1)  {  arr2[i] = arr1[i];  }  return 0;  } |





|  |
| --- |
| #include <stdio.h>  #define FOR(i, x, y) for(int i = x;i <=y; i++)  #define RE(i, x, y) for(int i = y; i>= x;i --)  #define maxN 100  int arr\_multiply(int \*arr1, int \*arr2, int arr\_size, int \*res);  int main()  {  int n = 3;  int a[maxN] = {1, 2, 3}, b[maxN] = {1, 2, 3};  int result[maxN] = {};  arr\_multiply(a, b, n, result);  FOR(i , 0, n-1)  {  printf("%d ",result[i]);  }  return 0;  }  /\*  ----------MULTIPLY ARRAYS---------------  Parameters:  - \*arr1 : the first array  - \*arr2 : the second array  - arr\_size : the size of the arrays  - \*res : the result array  Function:  - Multiplies each element of the first array with the corresponding element  in the second array (i.e., res[i] = arr1[i] \* arr2[i]), from index 0 to arr\_size - 1.  - The resulting values are stored in \*res. The ouput is 0 if the function not error.  \*/  int arr\_multiply(int \*arr1, int \*arr2, int arr\_size, int \*res)  {  FOR(i, 0, arr\_size - 1)  {  res[i] = arr1[i] \* arr2[i];  }  return 0;  } |



|  |
| --- |
| #include <stdio.h>  #define FOR(i, x, y) for(int i = x;i <= y; i++)  #define RE(i, x, y) for(int i = y; i >= x;i --)  #define maxN 100  int arr\_2d\_mutiply(int arr1[][maxN], int arr2[][maxN], int arr\_col, int arr\_row, int res[][maxN]);  int main()  {  int row = 3, col = 3;  int a[maxN][maxN] = {  {1, 2, 3},  {4, 5, 6},  {7, 8, 9}  }, b[maxN][maxN] = {  {1, 2, 3},  {4, 5, 6},  {7, 8, 9}  } ;  int result[maxN][maxN]={};  arr\_2d\_mutiply( a, b, col, row, result);  FOR(i, 0, row - 1)  {  FOR(j, 0, col - 1)  {  printf("%d ", result[i][j]);  }  printf("\n");  }  return 0;  }  /\*  ----------MULTIPLY 2D ARRAYS ---------------  Parameters:  - \*arr1 : the first array  - \*arr2 : the second array  - arr\_col : the colums of the arrays  - arr\_row : the rows of the arrays  - \*res : the result array  Function:  - Multiplies each element of the first array with the corresponding element  in the second array (i.e., res[i][j] = arr1[i][j] \* arr2[i][j]). The arr\_col and arr\_row indicate for the size include arr\_col colums and arr\_row rows of res.  - The resulting values are stored in \*res.  \*/  int arr\_2d\_mutiply(int arr1[][maxN], int arr2[][maxN], int arr\_col, int arr\_row, int res[][maxN]){  FOR(i, 0, arr\_row - 1)  {  FOR(j, 0, arr\_col - 1)  {  res[i][j] = arr1[i][j] \* arr2[i][j];  }  }  return 0;  } |



|  |
| --- |
| #include <stdio.h>  #include <string.h>  #define FOR(i, x, y) for(int i = x;i <=y; i++)  #define RE(i, x, y) for(int i = y; i>= x;i --)  #define maxN 1000  int char\_first(char \*s, char character);  int main ()  {  char c[maxN];  fgets(c, sizeof(c), stdin);  c[strlen(c)-1] = '\0';  FOR(i, 0, strlen(c) - 1)  {  if(!char\_first(c, c[i]))  {  printf("%c ", c[i]);  }  }  return 0;  }  /\*  ----------CHAR FIRST---------------  Paramaters:  - \*s : the string contain the character  - character : the character that you need to check  Function:  - The output is '1' if character appear 1 time on the string s  and output is '0 if the character appear more than 1 time. The ouput is 0 if the function not error.  \*/  int char\_first(char \*s, char character)  {  int cnt = 0;  FOR(i, 0, strlen(s) - 1)  {  if(s[i] == character)++ cnt;  if(cnt == 2) return 1;  }  return 0;  } |



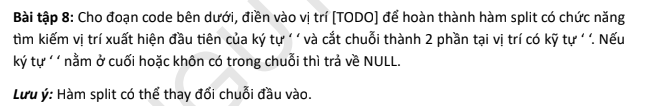
|  |
| --- |
| #include <stdio.h>  #include <string.h>  #define FOR(i, x, y) for(int i = x;i <=y; i++)  #define RE(i, x, y) for(int i = y; i>= x;i --)  #define maxN 1000  int char\_find\_a\_re\_b(char \*s,char char1, char char2);  int main ()  {  char c[maxN] , a, b;  scanf("%c %c", &a, &b);  getchar();  fgets(c, sizeof(c), stdin);  c[strlen(c) - 1] = '\0';  char\_find\_a\_re\_b(c, a, b);  printf("%s", c);  return 0;  }  /\*  ----------FIND THE CHAR A AND REPLACE BY THE CHAR B---------------  Paramaters:  - \*s : the string contain the character a  - char a : the character a  - char b : the character b  Function:  - find the char a in the string s and replace it by the char b. The ouput is 0 if the function not error.  \*/  int char\_find\_a\_re\_b(char \*s,char char1, char char2)  {  FOR(i, 0, strlen(s))  {  if(s[i] == char1)  {  s[i] = char2;  }  }  return 0;  } |



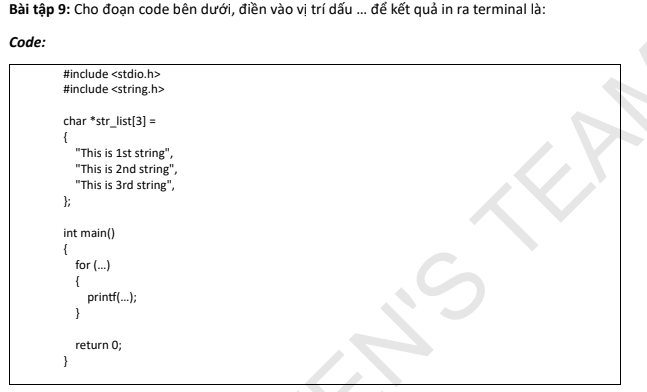
|  |
| --- |
| #include <stdio.h>  #include <string.h>  #define FOR(i, x, y) for(int i = x;i <=y; i++)  #define RE(i, x, y) for(int i = y; i>= x;i --)  #define maxN 1000  int char\_reverse(char \*s);  int main()  {  char c[maxN];  fgets(c, sizeof(c), stdin);  c[strlen(c) - 1] = '\0';  char\_reverse(c);  printf("%s", c);  return 0;  }  /\*  ----------FIND THE CHAR A AND REPLACE BY THE CHAR B---------------  Paramaters:  - \*s : the string  Function:  - The output is the revese of the string s . The ouput is 0 if the function not error.  \*/  int char\_reverse(char \*s)  {  char temp;  int left = 0;  int right = strlen(s) - 1;  while (left < right)  {  temp = s[left];  s[left] = s[right];  left ++;  right --;  }  return 0;  } |



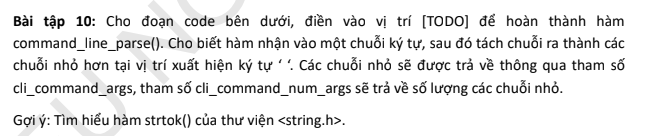
|  |
| --- |
| #include <stdio.h>  #include <string.h>  #define FOR(i, x, y) for(int i = x;i <=y; i++)  #define RE(i, x, y) for(int i = y; i>= x;i --)  #define maxN 1000  bool str\_cmp(char \*string1, char\* string2);  /\*  ----------COMPARING THE STRING---------------  Paramaters:  - \*string1 : the string 1  - \*string 2 : the string 2  Function:  - The output is '1' if string 1 and string 2 are equal  and output is '0 if string 1 and string 2 not equal  \*/  bool str\_cmp(char \*string1, char\* string2)  {  int n = strlen(string1);  int m = strlen(string2);  if( n == m)  {  FOR(i, 0, n - 1)  {  if(string1[i] != string2[i])  {  return 1;  }  }  }  else  {  return 1;  }  return 0;  }  int main ()  {  char c[maxN]="hello", p[maxN]="Hello";  if(str\_cmp(c, p))  {  printf("Error");  }  else  {  printf("Yup");  }  return 0;  } |



|  |
| --- |
| #include <stdio.h>  #include <string.h>  char \*split(char \*str)  {  // [TODO]  for(int i = 0; i < strlen(str); i++)  {  if(str[i] == ' ')  {  if( i == 0 || i == strlen(str) - 1)  {  return NULL;  }  else  {  str[i]='\0';  return &str[i+1];  }    }  }  return NULL;  }  int main()  {  char \*res\_str;  // Case 1:  char str\_1[] = "Hello World";  printf("\n=====\n");  printf("Case 1: str\_1 = %s\n", str\_1);  res\_str = split(str\_1);  printf("str\_1 = %s\n", str\_1);  if (res\_str != NULL)  printf("res\_str = %s\n", res\_str);  else  printf("No more string behind ' ' or string do not have ' '");  // Case 2:  char str\_2[] = "Hello ";  printf("\n=====\n");  printf("Case 2: str\_2 = %s\n", str\_2);  res\_str = split(str\_2);  printf("str\_2 = %s\n", str\_2);  if (res\_str != NULL)  printf("res\_str = %s\n", res\_str);  else  printf("No more string behind ' ' or string do not have ' '\n");  // Case 2:  char str\_3[] = "Hello ";  printf("\n=====\n");  printf("Case 3: str\_3 = %s\n", str\_3);  res\_str = split(str\_3);  printf("str\_3 = %s\n", str\_3);  if (res\_str != NULL)  printf("res\_str = %s\n", res\_str);  else  printf("No more string behind ' ' or string do not have ' '\n");  return 0;  } |



|  |
| --- |
| #include <stdio.h>  #include <string.h>  char \*str\_list[3] =  {  "This is 1st string",  "This is 2nd string",  "This is 3rd string",  };  int main()  {  for (int i = 0;i < 3; i++)  {  printf("%s\n", str\_list[i]);  }  return 0;  } |



|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdint.h>  #include <stdlib.h>  #define CLI\_COMMAND\_MAX\_NUM\_ARGS 10  /\*\*  \* @brief Split a command string into tokens using ' '  \*  \* @param[in] cli\_command\_buff The input command string (modifiable)  \* @param[in] cli\_buff\_size Length of the input buffer  \* @param[out] cli\_command\_args Array of char\* to store each token (must have enough space)  \* @param[out] cli\_command\_num\_args Pointer to store number of tokens  \*/  void command\_line\_parse(char \*cli\_command\_buff, uint8\_t cli\_buff\_size,  char \*\*cli\_command\_args, uint8\_t \*cli\_command\_num\_args)  {  char \*token = strtok(cli\_command\_buff, " ");  while(token != NULL){  cli\_command\_args[\*cli\_command\_num\_args] = token;  \*cli\_command\_num\_args = \*cli\_command\_num\_args + 1;  token = strtok(NULL, " ");  }  }  int main()  {  char command\_str[] = "SET GPIO A0 1";  char \*args\_list[CLI\_COMMAND\_MAX\_NUM\_ARGS]; // mảng con trỏ  uint8\_t num\_args = 0;  command\_line\_parse(command\_str, sizeof(command\_str), args\_list, &num\_args);  for (int i = 0; i < num\_args; i++) {  printf("Argument %d: %s\n", i, args\_list[i]);  }  return 0;  } |