Online C Compiler | CSA0275-SLOT_A.docx[1].pdf | ChatGPT

https://www.programiz.com/c-programming/online-compiler/

Programiz C Online Compiler

C Certification ›

main.c

Save Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NO_OF_CHARS 256
int max(int a, int b) {
    return (a > b) ? a : b;
}
int longestUniqueSubsttr(char *str) {
    int n = strlen(str);
    int cur_len = 1;
    int max_len = 1;
    int prev_index;
    int i;
    int *visited = (int *)malloc(sizeof(int) * NO_OF_CHARS);
    for (i = 0; i < NO_OF_CHARS; i++)
        visited[i] = -1;
    visited[str[0]] = 0;
    for (i = 1; i < n; i++) {
        prev_index = visited[str[i]];
        if (prev_index == -1 || i - cur_len > prev_index)
```

Output

Clear

```
/tmp/S94r2uVm96.o
The input string is: abcabcbb
The length of the longest non-repeating substring is: 3
```

32°C Haze

Search

ENG IN

12:38 22-02-2024

Online C Compiler | CSA0275-SLOT_A.docx[1].pdf | ChatGPT | Online C Compiler

https://www.programiz.com/c-programming/online-compiler/

Programiz C Online Compiler

C Certification ›

main.c

Save Run

Output Clear

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char* longestCommonPrefix(char* arr[], int n) {
    if (n == 0) return "";
    if (n == 1) return arr[0];
    qsort(arr, n, sizeof(char*), strcmp);
    int len = strlen(arr[0]);
    char* first = arr[0];
    char* last = arr[n - 1];
    int i = 0;
    while (i < len && first[i] == last[i]) {
        i++;
    }
    char* prefix = (char*)malloc(i + 1);
    strncpy(prefix, first, i);
    prefix[i] = '\0';

    return prefix;
}
```

```
/tmp/0qAU9Un0fd.o
The longest common prefix is: fl
```

Usman Road
Closed road

Search

ENG
IN

12:46
22-02-2024

C:\Users\Byalla Vishnu\Desktop\max product.cpp - [Executing] - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

max product.cpp

```
13
14      int maxEndingHere = arr[0];
15      int minEndingHere = arr[0];
16      int maxSoFar = arr[0];
17
18      for (int i = 1; i < n; i++) {
19          if (arr[i] < 0) {
20              int temp = maxEndingHere;
21              maxEndingHere = max(arr[i], arr[i] * minEndingHere);
22              minEndingHere = min(arr[i], arr[i] * temp);
23          } else {
24              maxEndingHere = max(arr[i], arr[i] * maxEndingHere);
25              minEndingHere = min(arr[i], arr[i] * minEndingHere);
26          }
27
28          maxSoFar = max(maxSoFar, maxEndingHere);
29      }
30
31      return maxSoFar;
32  }
33
34  int main() {
35      int arr[] = {2, 3, -2, 4};
36      int n = sizeof(arr) / sizeof(arr[0]);
37      int maxProduct = maxProductSubarray(arr, n);
38      printf("The maximum product of a subarray is %d\n", maxProduct);
39      return 0;
40  }
41
```

Compiler Resources Compile Log Debug Find Results Console Close

Abort Compilation

Shorten compiler pat

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Byalla Vishnu\Desktop\max product.exe
- Output Size: 323.6923828125 KiB
- Compilation Time: 0.28s
```

Line: 27 Col: 1 Sel: 0 Lines: 41 Length: 1005 Insert Done parsing in 0 second

C:\Users\Byalla Vishnu\Deskto

```
The maximum product of a subarray is 6

--------------------------------
Process exited after 3.018 seconds with return value 0
Press any key to continue . . .
```

Search

ENG IN

12:38 22-02-2024

C:\Users\Byalla Vishnu\Desktop\remove duplicate elements.c - [Executing] - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

max product.cpp | remove duplicate elements.c

```c
#include <stdio.h>
int removeDuplicates(int arr[], int n) {
    if (n == 0 || n == 1) return n;
    int j = 0;
    for (int i = 0; i < n - 1; i++) {
        if (arr[i] != arr[i + 1]) {
            arr[j++] = arr[i];
        }
    }
    arr[j++] = arr[n - 1];
    return j;
}
int main() {
    int arr[] = {1, 1, 2, 2, 3, 4, 4, 5, 6, 6};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    n = removeDuplicates(arr, n);
    printf("Array after removing duplicates:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

Compiler | Resources | Compile Log | Debug | Find Results | Console | Close

Abort Compilation

Shorten compiler pat

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Byalla Vishnu\Desktop\remove duplicate elements.
- Output Size: 323.5166015625 KiB
- Compilation Time: 0.33s
```

Line: 11 Col: 15 Sel: 0 Lines: 29 Length: 729 Insert Done parsing in 0 second

C:\Users\Byalla Vishnu\Deskt

```
Original sorted array:
1 1 2 2 3 4 4 5 6 6
Array after removing duplicates:
1 2 3 4 5 6

--------------------------------
Process exited after 3.484 seconds with return value 0
Press any key to continue . . .
```

Search

ENG IN

13:43 22-02-2024

main.c ×

Save ▷ Run

```c
#include <stdio.h>

// Function to find the index of the first occurrence of a target value in a sorted array
int binarySearchFirstOccurrence(int arr[], int size, int target) {
    int low = 0, high = size - 1;
    int result = -1;  // Initialize result to -1 (not found)

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == target) {
            // Update result and search towards the left for the first occurrence
            result = mid;
            high = mid - 1;
        } else if (arr[mid] < target) {
            // If target is greater, search in the right half
            low = mid + 1;
        } else {
            // If target is smaller, search in the left half
            high = mid - 1;
        }
    }

    return result;
}

int main() {
    int arr[] = {1, 2, 2, 4, 4, 4, 5, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
```

Output

```
$ /tmp/a.out
The first occurrence of 4 is at index 3
```

main.c ×

Save Run

```c
#include <stdio.h>

#define MAX_SIZE 100


void generateCombinations(int arr[], int data[], int start, int end, int index, int k);

// Function to print an array
void printArray(int arr[], int size);

int main() {
    int arr[] = {1, 2, 3, 4};
    int n = sizeof(arr) / sizeof(arr[0]);
    int k = 2; // Set the length of combinations

    int data[k]; // Temporary array to store combinations

    // Generate combinations
    generateCombinations(arr, data, 0, n - 1, 0, k);

    return 0;
}

void generateCombinations(int arr[], int data[], int start, int end, int index, int k) {
    // If a combination of length k is found, print it
    if (index == k) {
        printArray(data, k);
        return;
    }

    // Iterate over the array and generate combinations
    for (int i = start; i <= end && end - i + 1 >= k - index; i++) {
```

Output

```
/tmp/a.out
1 2
1 3
1 4
2 3
2 4
3 4
```

main.c ×

Save | Run

```c
#include <stdio.h>
#include <stdbool.h>

#define SIZE 9

// Function to check if a Sudoku board is valid
bool isValidSudoku(char board[SIZE][SIZE]) {
    // Check each row and column
    for (int i = 0; i < SIZE; i++) {
        // Arrays to keep track of seen digits in rows and columns
        int rowSet[SIZE] = {0};
        int colSet[SIZE] = {0};

        for (int j = 0; j < SIZE; j++) {
            // Check rows
            if (board[i][j] != '.' && rowSet[board[i][j] - '1'] == 1) {
                return false;  // Duplicate in the same row
            }
            rowSet[board[i][j] - '1'] = 1;

            // Check columns
            if (board[j][i] != '.' && colSet[board[j][i] - '1'] == 1) {
                return false;  // Duplicate in the same column
            }
            colSet[board[j][i] - '1'] = 1;
        }
    }

    // Check each 3x3 subgrid
    for (int block = 0; block < SIZE; block += 3) {
        for (int i = 0; i < SIZE; i += 3) {
```

Output

```
$ /tmp/a.out
The Sudoku board is valid.
```

main.c

Save Run

```
36      return 0;
37  }
38  void rotateByOne(int arr[])
39  {
40      int i, last;
41      last = arr[SIZE - 1];
42      for(i=SIZE-1; i>0; i--)
43      {
44          arr[i] = arr[i - 1];
45      }
46      arr[0] = last;
47  }
48
49
50  /**
51   * Print the given array
52   */
53  void printArray(int arr[])
54  {
55      int i;
56
57      for(i=0; i<SIZE; i++)
58      {
59          printf("%d ", arr[i]);
60      }
61  }
```

Output

Clear

```
/tmp/wY977cN9KA.o
Enter 10 elements array: 1
2
3
4
5
6
7
8
9
2
Enter number of times to right rotate: 10
Array before rotationn1 2 3 4 5 6 7 8 9 2

Array after rotation
1 2 3 4 5 6 7 8 9 2
```