Início	terça, 5 de abril de 2022 às 14:42
Estado	Prova submetida
Data de submissão:	quarta, 6 de abril de 2022 às 12:29
Tempo gasto	21 horas 46 minutos
Nota	**100** do máximo 100

Pergunta 1

Correta Pontuou 20 de 20

Write a C++ program to check whether a character, different from space, tab or newline, is a letter (uppercase or lowercase), a digit or another type of character.

The program must do the following:

1. read a character from the keyboard;
2. write on the screen the string `"LETTER"`, `"DIGIT"` or `"OTHER"` depending on the read character.

You can not use any library functions or classes, including C++ classes `string`, `stringstream`, `vector`, `list` or C library functions `sscanf` and `strtol`.

Por exemplo:

Entrada	Resultado
z	LETTER
X	LETTER
0	DIGIT
-	OTHER
!	OTHER

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

```
#include <iostream>
using namespace std;

int main(){
    char c;
    cin >> c;
    if ('0'<=c && c <= '9'){
        cout << "DIGIT" << endl;
    }
    else if(('A' <= c && c <= 'Z') || ('a' <= c && c <= 'z')){
        cout << "LETTER" << endl;
    }
    else{
        cout << "OTHER" <<  endl;
    }
    return 0;
}
```

	Entrada	Esperado	Recebido	
✔	z	LETTER	LETTER	✔
✔	X	LETTER	LETTER	✔
✔	0	DIGIT	DIGIT	✔
✔	-	OTHER	OTHER	✔
✔	!	OTHER	OTHER	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (Cpp):

```
#include <iostream>
using namespace std;

int main(void) {
  //cout << "One char ? "
  char c;
```

```
  cin >> c;
  if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z'))
    cout << "LETTER";
  else if (c >= '0' && c <= '9')
    cout << "DIGIT";
  else
    cout << "OTHER";
  cout << endl;
  return 0;
}

// private tests (1000 points each)
// 'g' => LETTER
// 'L' => LETTER
// '8' => DIGIT
// '\' => OTHER
```

Correta

Nota desta submissão: 20/20

Pergunta 2

Correta Pontuou 20 de 20

Write a C++ function `void remove_duplicates(int a[], int& size)` that takes as parameters an array of positive integers, `a`, sorted in ascending order, and its number of elements, `size`, and removes all the duplicate elements, keeping the result in the same array. This can be achieved by moving the non-duplicates to the first elements of the array and updating the number of elements (parameter `size`).

Por exemplo:

Teste	Resultado

`int a[] = { 1, 2, 3 };`
`int size = sizeof(a) / sizeof(int);`
`remove_duplicates(a, size);`
`print(a, size);`
	`[ 1 2 3 ]`

`int a[] = { 1, 1, 2, 2, 3, 3, 3 };`
`int size = sizeof(a) / sizeof(int);`
`remove_duplicates(a, size);`
`print(a, size);`
	`[ 1 2 3 ]`

`int a[] = { 1, 1, 1, 1, 2, 3, 3 };`
`int size = sizeof(a) / sizeof(int);`
`remove_duplicates(a, size);`
`print(a, size);`
	`[ 1 2 3 ]`

`int a[] = { 1, 1, 1, 1, 2, 2, 2, 2, 33 };`
`int size = sizeof(a) / sizeof(int);`
`remove_duplicates(a, size);`
`print(a, size);`
	`[ 1 2 33 ]`

`int a[] = { 1, 1 };`
`int size = sizeof(a) / sizeof(int);`
`remove_duplicates(a, size);`
`print(a, size);`
	`[ 1 ]`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
#include <iostream>
using namespace std;

//! Print array.
void print(int a[], int size) {
  cout << "[ ";
  for (int i = 0; i < size; i++)
    cout << a[i] << " ";
  cout << "]";
  cout << endl;
}

void remove_duplicates(int a[], int& size){
    int newsize = size;
    int p = 1;
    int n = 1;
    while(n < size){
        if (a[n] != a[n-1]){
            a[p] = a[n];
            n++; p++;
        }
        else{
```

	Teste	Esperado	Recebido	

✔	`int a[] = { 1, 2, 3 };`
	`int size = sizeof(a) / sizeof(int);`
	`remove_duplicates(a, size);`
	`print(a, size);`
		`[ 1 2 3 ]`	`[ 1 2 3 ]`	✔

	Teste	Esperado	Recebido	

✔	`int a[] = { 1, 1, 2, 2, 3, 3, 3 };`
	`int size = sizeof(a) / sizeof(int);`
	`remove_duplicates(a, size);`
	`print(a, size);`
		`[ 1 2 3 ]`	`[ 1 2 3 ]`	✔

✔	`int a[] = { 1, 1, 1, 1, 2, 3, 3 };`
	`int size = sizeof(a) / sizeof(int);`
	`remove_duplicates(a, size);`
	`print(a, size);`
		`[ 1 2 3 ]`	`[ 1 2 3 ]`	✔

✔	`int a[] = { 1, 1, 1, 1, 2, 2, 2, 2, 2, 33 };`
	`int size = sizeof(a) / sizeof(int);`
	`remove_duplicates(a, size);`
	`print(a, size);`
		`[ 1 2 33 ]`	`[ 1 2 33 ]`	✔

✔	`int a[] = { 1, 1 };`
	`int size = sizeof(a) / sizeof(int);`
	`remove_duplicates(a, size);`
	`print(a, size);`
		`[ 1 ]`	`[ 1 ]`	✔

✔	`int a[] = { 10, 10, 10, 20, 20, 20, 20, 20, 20, 30, 30, 30, 40, 40, 40, 500 };`
	`int size = sizeof(a) / sizeof(int);`
	`remove_duplicates(a, size);`
	`print(a, size);`
		`[ 10 20 30 40 500 ]`	`[ 10 20 30 40 500 ]`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <iostream>
using namespace std;

//! Print array.
void print(int a[], int size) {
  cout << "[ ";
  for (int i = 0; i < size; i++)
    cout << a[i] << " ";
  cout << "]";
  cout << endl;
}

//! Removes all the duplicate elements.
void remove_duplicates(int a[], int& size) {
  int i = 0;
  while (i < size - 1) {
    if (a[i] == a[i + 1]) {
      for (int j = i; j < size - 1; j++) {
        a[j] = a[j + 1];
      }
      size--;
    }
```

Correta

Nota desta submissão: 20/20

Pergunta 3

Correta Pontuou 20 de 20

Two character sequences are anagrams of one another if they contain exactly the same letters and letter counts, even if in a different order. For instance, “Amor” and “Roma” are (portuguese) anagrams.

Write a C++ function `bool anagrams(const char a[], const char b[], int& n)` such that:

- it takes as input C strings `a` and `b` that can contain uppercase or lowercase letter characters (‘A’ to ‘Z’, ‘a’ to ‘z’) and also the space character (‘ ‘), and a reference `int` argument `n`;
- return `true` if `a` and `b` are anagrams of one another, and `false` otherwise; and
- on exit, the value of `n` should contain the total number of letters that are different between both strings (this will be `0` if and only if the strings are anagrams).

Hint: there are 26 letters in the alphabet. Use an internal array of length 26 to compute the difference of occurrences per each letter between both strings.

You cannot use any library classes or functions, including `vector`, `list`, `string`, `qsort` and `sort`.

Por exemplo:

Teste	Resultado

`char a[] = "";`
`char b[] = "";`
`int n = -1;`
`bool r = anagrams(a, b, n);`
`cout << '\"' << a << '\"' << ' '`
`  << '\"' << b << '\"' << ' '`
`  << boolalpha << r << ' ' << n << '\n';`
	`"" "" true 0`

`char a[] = " R o m a ";`
`char b[] = "Amor";`
`int n = -1;`
`bool r = anagrams(a, b, n);`
`cout << '\"' << a << '\"' << ' '`
`  << '\"' << b << '\"' << ' '`
`  << boolalpha << r << ' ' << n << '\n';`
	`" R o m a " "Amor" true 0`

`char a[] = "Amor a";`
`char b[] = "Roma";`
`int n = -1;`
`bool r = anagrams(a, b, n);`
`cout << '\"' << a << '\"' << ' '`
`  << '\"' << b << '\"' << ' '`
`  << boolalpha << r << ' ' << n << '\n';`
	`"Amor a" "Roma" false 1`

`char a[] = " amor e ";`
`char b[] = "ROSAS";`
`int n = -1;`
`bool r = anagrams(a, b, n);`
`cout << '\"' << a << '\"' << ' '`
`  << '\"' << b << '\"' << ' '`
`  << boolalpha << r << ' ' << n << '\n';`
	`" amor e " "ROSAS" false 4`

`char a[] = "Z Plus Plus after Python";`
`char b[] = "zYTHON after p plus plus";`
`int n = -1;`
`bool r = anagrams(a, b, n);`
`cout << '\"' << a << '\"' << ' '`
`  << '\"' << b << '\"' << ' '`
`  << boolalpha << r << ' ' << n << '\n';`
	`"Z Plus Plus after Python" "zYTHON after p plus plus" true 0`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

```
bool anagrams(const char a[], const char b[], int& n){
    int aa[26] = {0};
    int bb[26] = {0};
    int i = 0;
    while(a[i] != '\0'){
        char c = a[i];
        if ('a' <= c && c <= 'z'){
            int index = c-'a';
            aa[index]++;
        }
        if ('A' <= c && c <= 'Z'){
```

```
            int index = c-'A';
            aa[index]++;
        }
        i++;
    }
    i = 0;
    while(b[i] != '\0'){
        char c = b[i];
        if ('a' <= c && c <= 'z'){
            int index = c-'a';
            bb[index]++;
```

	Teste	Esperado	Recebido	

✔	`char a[] = "";`
	`char b[] = "";`
	`int n = -1;`
	`bool r = anagrams(a, b, n);`
	`cout << '\"' << a << '\"' << ' '`
	`  << '\"' << b << '\"' << ' '`
	`  << boolalpha << r << ' ' << n << '\n';`
		`"" "" true 0`	`"" "" true 0`	✔

✔	`char a[] = " R o m a ";`
	`char b[] = "Amor";`
	`int n = -1;`
	`bool r = anagrams(a, b, n);`
	`cout << '\"' << a << '\"' << ' '`
	`  << '\"' << b << '\"' << ' '`
	`  << boolalpha << r << ' ' << n << '\n';`
		`" R o m a " "Amor" true 0`	`" R o m a " "Amor" true 0`	✔

✔	`char a[] = "Amor a";`
	`char b[] = "Roma";`
	`int n = -1;`
	`bool r = anagrams(a, b, n);`
	`cout << '\"' << a << '\"' << ' '`
	`  << '\"' << b << '\"' << ' '`
	`  << boolalpha << r << ' ' << n << '\n';`
		`"Amor a" "Roma" false 1`	`"Amor a" "Roma" false 1`	✔

✔	`char a[] = " amor e ";`
	`char b[] = "ROSAS";`
	`int n = -1;`
	`bool r = anagrams(a, b, n);`
	`cout << '\"' << a << '\"' << ' '`
	`  << '\"' << b << '\"' << ' '`
	`  << boolalpha << r << ' ' << n << '\n';`
		`" amor e " "ROSAS" false 4`	`" amor e " "ROSAS" false 4`	✔

	Teste	Esperado	Recebido	

✔	char a[] = "Z Plus Plus after Python";
	char b[] = "zYTHON after p plus plus";
	int n = -1;
	bool r = anagrams(a, b, n);
	cout << '\"' << a << '\"' << ' '
	     << '\"' << b << '\"' << ' '
	     << boolalpha << r << ' ' << n << '\n';
		"Z Plus Plus after Python" "zYTHON after p plus plus" true 0	"Z Plus Plus after Python" "zYTHON after p plus plus" true 0	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
1  //! Determines if two s-trings are anagrams.
2  bool anagrams(const char a[], const char b[], int& n) {
3    int count[26] = { 0 };
4    // add letters of a
5    for (int i = 0; a[i] != '\0'; i++) {
6      if (a[i] != ' ') {
7        if (a[i] >= 'a' && a[i] <= 'z')
8          count[a[i] - 'a']++;
9        else
10         count[a[i] - 'A']++;
11     }
12   }
13   // remove letters of b
14   for (int i = 0; b[i] != '\0'; i++) {
15     if (b[i] != ' ') {
16       if (b[i] >= 'a' && b[i] <= 'z')
17         count[b[i] - 'a']--;
18       else
19         count[b[i] - 'A']--;
20     }
21   }
22   // count different letters
```

Correta

Nota desta submissão: 20/20

Pergunta 4

Correta Pontuou 20 de 20

The Mandelbrot set recurrence $z_0, z_1, z_2, \ldots$ of complex numbers is defined for a complex constant c by:

$z_0 = 0$

$z_n = z_{n-1} \times z_{n-1} + c \quad \forall n > 0$

Consider the type `complex` to represent complex numbers as follows:

```
struct complex {
  int real;  // Real part
  int img;   // Imaginary part
};
```

Note that, for simplicity of calculations, both the real and imaginary parts of complex numbers considered here are integers.

Write a C++ function `void mandel(complex c, int n, complex z[])` that returns in array `z` the first `n` complex numbers (z_0 to z_{n-1}) for the Mandelbrot set recurrence defined for `c`.

You can not use `pow` or other functions defined in `cmath` ou `math.h`.

Hints:

- Recall that for two complex numbers $c_1 = x_1 + y_1 i$ and $c_2 = x_2 + y_2 i$ we have that $c_1 + c_2 = (x_1 + x_2) + (y_1 + y_2)i$ and $c_1 \times c_2 = (x_1 x_2 - y_1 y_2) + (x_1 y_2 + x_2 y_1)\, i$
- The problem should become simpler to solve if you start by defining auxiliary functions to implement the sum and the multiplication of complex numbers.
- Note that all elementary calculations involve integer numbers. Do not use floating point arithmetic: it is not required and it will not help you!

Por exemplo:

Teste	Resultado

`complex c = { 0, 0 };`
`const int n = 1;`
`complex z[n];`
`mandel(c, n, z);`
`print(z, n);`
	`[ 0 ]`

`complex c = { 0, 0 };`
`const int n = 3;`
`complex z[n];`
`mandel(c, n, z);`
`print(z, n);`
	`[ 0 0 0 ]`

`complex c = { 1, 1 };`
`const int n = 3;`
`complex z[n];`
`mandel(c, n, z);`
`print(z, n);`
	`[ 0 1+1i 1+3i ]`

`complex c = { -1, 0 };`
`const int n = 6;`
`complex z[n];`
`mandel(c, n, z);`
`print(z, n);`
	`[ 0 -1 0 -1 0 -1 ]`

`complex c = { 0, 3 };`
`const int n = 5;`
`complex z[n];`
`mandel(c, n, z);`
`print(z, n);`
	`[ 0 +3i -9+3i 72-51i 2583-7341i ]`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
#include <iostream>
using namespace std;

//! Complex number
struct complex {
  int real;  // Real part
  int img;   // Imaginary part
};
```

```
 9
10  //! Print array of complex numbers.
11  void print(const complex z[], int n) {
12    cout << "[ ";
13    for (int i = 0; i < n; i++) {
14      if (z[i].real == 0 && z[i].img == 0)
15        cout << 0;
16      else {
17        if (z[i].real != 0)
18          cout << z[i].real;
19        if (z[i].img > 0)
20          cout  << '+' << z[i].img << 'i';
21        else if (z[i].img < 0)
22          cout  << z[i].img << 'i';
```

	Teste	Esperado	Recebido	

✔	`complex c = { 0, 0 };`
	`const int n = 1;`
	`complex z[n];`
	`mandel(c, n, z);`
	`print(z, n);`
		`[ 0 ]`	`[ 0 ]`	✔

✔	`complex c = { 0, 0 };`
	`const int n = 3;`
	`complex z[n];`
	`mandel(c, n, z);`
	`print(z, n);`
		`[ 0 0 0 ]`	`[ 0 0 0 ]`	✔

✔	`complex c = { 1, 1 };`
	`const int n = 3;`
	`complex z[n];`
	`mandel(c, n, z);`
	`print(z, n);`
		`[ 0 1+1i 1+3i ]`	`[ 0 1+1i 1+3i ]`	✔

✔	`complex c = { -1, 0 };`
	`const int n = 6;`
	`complex z[n];`
	`mandel(c, n, z);`
	`print(z, n);`
		`[ 0 -1 0 -1 0 -1 ]`	`[ 0 -1 0 -1 0 -1 ]`	✔

✔	`complex c = { 0, 3 };`
	`const int n = 5;`
	`complex z[n];`
	`mandel(c, n, z);`
	`print(z, n);`
		`[ 0 +3i -9+3i 72-51i 2583-7341i ]`	`[ 0 +3i -9+3i 72-51i 2583-7341i ]`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  //! Complex number
 5  struct complex {
 6    int real;  // Real part
 7    int img;   // Imaginary part
 8  };
 9
10  //! Print array of complex numbers.
11  void print(const complex z[], int n) {
12    cout << "[ ";
13    for (int i = 0; i < n; i++) {
14      if (z[i].real == 0 && z[i].img == 0)
15        cout << 0;
16      else {
17        if (z[i].real != 0)
18          cout << z[i].real;
19        if (z[i].img > 0)
20          cout  << '+' << z[i].img << 'i';
21        else if (z[i].img < 0)
22          cout  << z[i].img << 'i';
```

Correta

Nota desta submissão: 20/20

Pergunta 5

Correta Pontuou 20 de 20

Consider the code given in node.cpp containing the definition of type `node`, supporting the definition of doubly-linked lists with `int` values, and associated functions:

- `node* build(int v, node* n)`: builds a new node with value `v` (the `value` member), followed by `n` (the `next` member) — if `n != nullptr` then `n->prev` is set to point to the new node;
- `void destroy(node* n)`: releases the memory allocated to `n` and successor nodes; and
- `void print(const node* n)`: prints values in the node pointed by `n`.

Define a new function `node* reverse(const node* n)` that returns a new list with the same elements as `n` in reverse order.

You need to include `node.cpp` file in your code i.e. `#include "node.cpp"`.

You cannot use any C++ library classes or functions, including `vector`, `list`, or `string`.

Por exemplo:

Teste	Resultado

`node* a = nullptr;`
`node* b = reverse(a);`
`destroy(a); destroy(b);`	

`node* a = build(1, nullptr);`
`node* b = reverse(a);`
`print(b);`
`destroy(a); destroy(b);`
	`(\<1<\)`

`node* a = build(1, build(2, nullptr));`
`node* b = reverse(a);`
`print(b);`
`destroy(a); destroy(b);`
	`(\<2<1)(2<1<\)`

`node* a = build(1, build(2, build(3, nullptr)));`
`node* b = reverse(a);`
`print(b);`
`destroy(a); destroy(b);`
	`(\<3<2)(3<2<1)(2<1<\)`

`node* a = build(1, build(2, build(3,build(4, build(5, nullptr)))));`
`node* b = reverse(a);`
`print(b);`
`destroy(a); destroy(b);`
	`(\<5<4)(5<4<3)(4<3<2)(3<2<1)(2<1<\)`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
#include "node.cpp"

node* reverse(const node* n){
    if(n == nullptr) return nullptr;
    while(n->next != nullptr)
        n = n->next;
    node* result = build(n->value, nullptr);
    node* first = result;
    while(n->prev != nullptr){
        n = n->prev;
        node* newnode = build(n->value, nullptr);
        newnode->prev = result;
        result->next = newnode;
        result = result->next;
    }
    return first;
}
```

	Teste	Esperado	Recebido	

	Teste	Esperado	Recebido	

✔	`node* a = nullptr;`
	`node* b = reverse(a);`
	`destroy(a); destroy(b);`
				✔

✔	`node* a = build(1, nullptr);`
	`node* b = reverse(a);`
	`print(b);`
	`destroy(a); destroy(b);`
		(\<1<\)	(\<1<\)	✔

✔	`node* a = build(1, build(2, nullptr));`
	`node* b = reverse(a);`
	`print(b);`
	`destroy(a); destroy(b);`
		(\<2<1)(2<1<\)	(\<2<1)(2<1<\)	✔

✔	`node* a = build(1, build(2, build(3, nullptr)));`
	`node* b = reverse(a);`
	`print(b);`
	`destroy(a); destroy(b);`
		(\<3<2)(3<2<1)(2<1<\)	(\<3<2)(3<2<1)(2<1<\)	✔

✔	`node* a = build(1, build(2, build(3,build(4,`
	`build(5, nullptr)))));`
	`node* b = reverse(a);`
	`print(b);`
	`destroy(a); destroy(b);`
		(\<5<4)(5<4<3)(4<3<2)
		(3<2<1)(2<1<\)
			(\<5<4)(5<4<3)(4<3<2)
			(3<2<1)(2<1<\)
				✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```c
#include "node.cpp"

node* reverse(const node* n) {
  if (n == nullptr) {
    return nullptr;
  }
  node* r = new node { n->value, nullptr, nullptr };
  n = n->next;
  while (n != nullptr) {
    r->prev = new node { n->value, nullptr, r };
    r = r->prev;
    n = n->next;
  }
  return r;
}
/*
  // private tests (1000 points each)
  {
    node* a = build(11, build(12, build(13, build(14))));
    node* b = reverse(a);
    print(b);
    destroy(a); destroy(b);
```

Correta

Nota desta submissão: 20/20

◄ T04 29/03

Ir para...

T05 05/04 ►