Losowanie liczb

#include <cstdlib>

#include <ctime>

#include <iostream>

int main()

{

    std::srand(std::time(nullptr));

    int computer = 2 \* (std::rand() % 2) - 1, player;

    std::cin >> player;

    std::cout << computer << std::endl;

    std::cout << std::boolalpha << (0 < computer \* player) << std::endl;

}

Ustawianie szerokości

for (int column = 1; column <= number; ++column)

        {

            std::cout << " " << std::setw(width) << row \* column;

        }

Wczytywanie przez cały czas

#include <iostream>

int main()

{

    int hour, minute;

    if (std::cin >> hour >> minute)

    {

        int sum = 0, start = 60 \* hour + minute;

        int count = 0;

        while (std::cin >> hour >> minute)

        {

            int stop = 60 \* hour + minute;

            sum += stop - start;

            ++count;

            start = stop;

        }

        if (count)

        {

            std::cout << double(sum) / count << std::endl;

        }

    }

}

 for (double value; std::cin >> value;)

    {

        sum += value;

        product \*= value;

        ++count;

    }

Referencje w funkcjach

void clock(double &small, double &large, int hour, int minute)

{

    small = (hour % 12) \* 30. + minute / 2.;

    large = minute \* 6.;

}

Zwracanie więcej niż jednej zmiennej przez referencje

double quadratic(double &x1, double &x2, double a, double b, double c)

{

    double delta = b \* b - 4 \* a \* c;

    if (delta >= 0)

    {

        x1 = (-b - std::sqrt(delta)) / a / 2;

        x2 = (-b + std::sqrt(delta)) / a / 2;

    }

    return delta;

}

W wektorach

Swap, push\_back dodaje elementy, resize roszerza pojemnik, też odwracanie wektora jest w Lab 5

Size()

Argumenty

int main(int argc, char \*argv[])

{

    double speed = std::atof(argv[1]);

 int limit = std::atoi(argv[1]);

Duże litery I pliki

#include <fstream>

char upper(char character)

{

    return 'a' <= character && character <= 'z' ? 'A' + character - 'a' : character;

}

int main(int argc, char \*argv[])

{

    std::ifstream input(argv[1]);

    std::ofstream output(argv[2]);

    char character;

    while (input.get(character))

    {

        output << upper(character);

    }

    output.close();

    input.close();

}

Zliczanie znaków, lini i wyrazów

#include <cctype>

#include <fstream>

#include <iostream>

int main(int argc, char \*argv[])

{

    std::ifstream input(argv[1]);

    int lines = 0, words = 0, characters = 0;

    for (char previous = '\n', current; input.get(current); previous = current)

    {

        if (previous == '\n')

        {

            ++lines;

        }

        if (std::isspace(previous) && !std::isspace(current))

        {

            ++words;

        }

        ++characters;

    }

    input.close();

    std::cout << lines << " " << words << " " << characters << std::endl;

}

Zamiana wyrazów znakami

#include <fstream>

#include <string>

int main(int argc, char \*argv[])

{

    std::string source = argv[1], target = argv[2];

    std::ifstream input(argv[3]);

    std::ofstream output(argv[4]);

    for (std::string line; std::getline(input, line);)

    {

        for (int index = 0;

             (index = line.find(source, index)) != std::string::npos;

             index += target.size())

        {

            line.replace(index, source.size(), target);

        }

        output << line << std::endl;

    }

    output.close();

    input.close();

}

Trymowanie tekstu

#include <fstream>

#include <string>

int main(int argc, char \*argv[])

{

    std::ifstream input(argv[1]);

    std::ofstream output(argv[2]);

    for (std::string line; std::getline(input, line); output << line << std::endl)

    {

        int index = line.find\_last\_not\_of(" \t");

        line.erase(index == std::string::npos ? 0 : index + 1);

    }

    output.close();

    input.close();

}

Strumienie

Czytanie rachunku i dodawanie tyko liczb

#include <fstream>

#include <iostream>

#include <sstream>

#include <string>

int main(int argc, char \*argv[])

{

    std::ifstream input(argv[1]);

    double sum = 0.;

    for (std::string line; std::getline(input, line);)

    {

        std::string word;

        for (std::istringstream stream(line); stream >> word;)

            ;

        double price;

        if (std::istringstream(word) >> price)

        {

            sum += price;

        }

    }

    std::cout << sum << std::endl;

    input.close();

}

Zadania z iteratorów

Usuwanie elementów poprzez przeniesienie i obcięcie

#include <iostream>

#include <vector>

std::vector<int>::iterator remove(std::vector<int>::iterator begin,

                                  std::vector<int>::iterator end,

                                  int element)

{

    for (auto iterator = begin; iterator < end; ++iterator)

    {

        if (\*iterator != element)

        {

            \*begin++ = \*iterator;

        }

    }

    return begin;

}

int main()

{

    std::vector<int> vector{-7, 5, 2, 2, 11, 2, 3};

    auto result = remove(vector.begin(), vector.end(), 2);

    vector.erase(result, vector.end());

    for (auto iterator = vector.cbegin(); iterator < vector.cend();)

    {

        std::cout << \*iterator++ << " ";

    }

    std::cout << std::endl;

}

Sortowanie przez selekcje

#include <iostream>

#include <utility>

#include <vector>

void selection\_sort(std::vector<int>::iterator b,std::vector<int>::iterator e)

 {

    while(b<e) {

        auto m=b;

        for (auto i=b;++i<e;)

        {

            if (\*i < \*m)

             {

                m=i;

             }

        }

        std::iter\_swap(b++, m);

        }}

int main() {

    std::vector<int> v {13, -2, 21, 5, -8, 5, 7, -10};

    selection\_sort(v.begin()+2,v.end()-2);

    for(auto i = v.begin();i<v.end();++i)

    {

        std::cout<<\*i<<" ";

    }

    std::cout<<std::endl;

    }

Jak działa getline

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Iterator jest używany w C++ do iteracji po elementach kontenerów, takich jak string. Pozwala na dostęp do kolejnych elementów w sekwencji i manipulację nimi. W przypadku stringów istnieją różne metody iteratorów, które można wykorzystać. Poniżej przedstawiam kilka przykładów:

1. **begin()** - Ta metoda zwraca iterator wskazujący na pierwszy element w stringu.

cppCopy code

std::string str = "Hello"; std::string::iterator it = str.begin();

1. **end()** - Ta metoda zwraca iterator wskazujący na fikcyjny element, który jest poza końcem stringu. Jest to tzw. "past-the-end" iterator.

cppCopy code

std::string str = "Hello"; std::string::iterator it = str.end();

1. **rbegin()** - Ta metoda zwraca iterator wskazujący na ostatni element w stringu. Pozwala na iterację od końca.

cppCopy code

std::string str = "Hello"; std::string::reverse\_iterator it = str.rbegin();

1. **rend()** - Ta metoda zwraca iterator wskazujący na fikcyjny element, który jest przed początkiem stringu. Jest to tzw. "past-the-begin" iterator dla iteracji od końca.

cppCopy code

std::string str = "Hello"; std::string::reverse\_iterator it = str.rend();

1. Przykład iteracji za pomocą iteratora:

cppCopy code

std::string str = "Hello"; for (std::string::iterator it = str.begin(); it != str.end(); ++it) { std::cout << \*it << " "; }

1. Przykład iteracji od końca za pomocą iteratora:

cppCopy code

std::string str = "Hello"; for (std::string::reverse\_iterator it = str.rbegin(); it != str.rend(); ++it) { std::cout << \*it << " "; }

1. Inne przydatne metody, które można używać w połączeniu z iteratorami stringów, to **++it** (przesunięcie iteratora do następnego elementu), **--it** (przesunięcie iteratora do poprzedniego elementu), **\*it** (dostęp do wartości wskazywanej przez iterator), itd.

Iteratorzy w C++ są potężnym narzędziem do manipulacji sekwencjami danych, takimi jak stringi. Pozwalają na dostęp do poszczególnych elementów i wykonanie różnych operacji na nich. Należy jednak pamiętać o odpowiednim zarządzaniu iteratorami, aby uniknąć wyjścia poza zakres lub błędów.

Oto kilka innych przydatnych metod, które można używać z iteratorami dla stringów w języku C++:

1. **advance(it, n)** - Ta funkcja przesuwa iterator **it** o **n** pozycji do przodu. Może być przydatna, jeśli chcemy przesunąć iterator o więcej niż jeden element.

std::string str = "Hello"; std::string::iterator it = str.begin(); std::advance(it, 2); // Przesuń iterator o 2 pozycje do przodu

1. **distance(first, last)** - Ta funkcja zwraca liczbę elementów pomiędzy iteratorami **first** i **last**. Może być używana do obliczenia odległości między dwoma iteratorami.

cppCopy code

std::string str = "Hello"; std::string::iterator first = str.begin(); std::string::iterator last = str.end(); int distance = std::distance(first, last);

1. **insert(it, value)** - Ta metoda wstawia wartość **value** przed pozycją wskazywaną przez iterator **it**. Przesuwa wszystkie elementy od tego miejsca do końca o jedną pozycję do przodu.

cppCopy code

std::string str = "Helo"; std::string::iterator it = str.begin(); str.insert(it, 'l'); // Wstaw 'l' przed pierwszym elementem

1. **erase(it)** - Ta metoda usuwa element wskazywany przez iterator **it** i zwraca iterator do następnego elementu. Przesuwa pozostałe elementy o jedną pozycję do tyłu.

cppCopy code

std::string str = "Hello"; std::string::iterator it = str.begin() + 2; std::string::iterator next = str.erase(it); // Usuń trzeci element ('l')

1. **replace(it, it2, value)** - Ta metoda zamienia wartość w zakresie pomiędzy iteratorami **it** i **it2** na wartość **value**. Może być przydatna do zamiany fragmentu stringa na inną wartość.

cppCopy code

std::string str = "Hello, World!"; std::string::iterator it = str.begin() + 7; std::string::iterator it2 = str.begin() + 12; str.replace(it, it2, "OpenAI"); // Zamień fragment "World" na "OpenAI"

1. **find(value)** - Ta metoda wyszukuje pierwsze wystąpienie wartości **value** w stringu i zwraca iterator do tego miejsca. Jeśli nie znaleziono wartości, zwracany jest iterator wskazujący na **end()**.

cppCopy code

std::string str = "Hello, World!"; std::string::iterator it = str.find("World");

Te metody stanowią tylko część dostępnych funkcji i operacji, które można wykonywać z iteratorami stringów w C++. Istnieje wiele innych metod, takich jak **substr()**, **compare()**, **reverse()**, które mogą być używane w połączeniu z iteratorami do manipulacji stringami.