///Izveidot C++ funkciju, kas ļauj noskaidrot, cik reizes augsta līmeņa simbolu virknē ir sastopams konkrēts simbols.

#include <iostream>

#include <string>

using namespace std;

int skaititSimbolu(const string& virkne, char mekletaisSimbols) {

    int skaititajs = 0;

    for (char simbols : virkne) {

        if (simbols == mekletaisSimbols) {

            skaititajs++;

        }

    }

    return skaititajs;

}

int main() {

    string teksts;

    char simbols;

    cout << "Ievadiet tekstu: ";

    getline(cin, teksts);

    cout << "Ievadiet meklējamo simbolu: ";

    cin >> simbols;

    int skaits = skaititSimbolu(teksts, simbols);

    cout << "Simbols '" << simbols << "' parādās " << skaits << " reizes." << endl;

    return 0;

}

///Sastādīt C++ klasi ”Punkts” izmantojot objektorientētās programmēšanas līdzekļus

// konstruktors; destruktors;

// setPunkts(x, y) – novieto objektu punktā ar koordinātēm x un y;

// getPunkts(x, y) – atgriež objekta koordinātes parametros x un y;

// increasePunkts(d) – palielina katras kooordinātes vērtību par doto vērtību d;

// print() – izdrukā objekta koordinātes.

// Punkts.h

class Punkts {

public:

    Punkts(); // Konstruktors

    ~Punkts(); // Destruktors

    void setPunkts(int x, int y);

    void getPunkts(int &x, int &y) const;

    void increasePunkts(int d);

    void print() const;

private:

    int x, y;

};

// Punkts.cpp

#include "Punkts.h"

#include <iostream>

using namespace std;

Punkts::Punkts() : x(0), y(0) {}

Punkts::~Punkts() {}

void Punkts::setPunkts(int x, int y) {

    this->x = x;

    this->y = y;

}

void Punkts::getPunkts(int &x, int &y) const {

    x = this->x;

    y = this->y;

}

void Punkts::increasePunkts(int d) {

    x += d;

    y += d;

}

void Punkts::print() const {

    cout << "Punkts(" << x << ", " << y << ")" << endl;

}

// main.cpp

#include "Punkts.h"

#include <iostream>

using namespace std;

int main() {

    // Izveido objektu tiešā veidā

    Punkts p1;

    p1.setPunkts(5, 10);

    p1.print();

    // Izveido objektu dinamiskā veidā

    Punkts \*p2 = new Punkts();

    p2->setPunkts(15, 20);

    p2->print();

    p1.increasePunkts(3);

    p1.print();

    p2->increasePunkts(3);

    p2->print();

    int x, y;

    p1.getPunkts(x, y);

    cout << "P1 koordinātes: (" << x << ", " << y << ")" << endl;

    p2->getPunkts(x, y);

    cout << "P2 koordinātes: (" << x << ", " << y << ")" << endl;

    delete p2;

    return 0;

}

Izveidot C++ programmu, kas izveido saīsinātu teksta rindiņu, ņemot tikai katru otro simbolu, sākot ar rindiņas pirmo simbolu. Tātad no “123 56” ir jāiegūst “135”.

Teksta rindiņu ievada lietotājs. Gan dotā teksta rindiņa, gan izveidotā teksta rindiņa jāsaglabā programmā kā zema līmeņa simbolu virknes.

Dotās teksta rindiņas apstrādei sastādīt funkciju, kura no saņemtās rindiņas izveido jaunu rindiņu un atgriež norādi uz jauno rindiņu.

#include <iostream>

#include <cstring>

using namespace std;

char\* saisinatTekstu(const char\* ievadTeksts) {

    int garums = strlen(ievadTeksts);

    char\* saisinats = new char[garums / 2 + 1]; // +1 for null terminator

    for (int i = 0, j = 0; i < garums; i += 2, j++) {

        saisinats[j] = ievadTeksts[i];

    }

    saisinats[garums / 2] = '\0'; // Pievieno beigu simbolu

    return saisinats;

}

int main() {

    const int MAX\_GARUMS = 1000;

    char teksts[MAX\_GARUMS];

    cout << "Ievadiet tekstu: ";

    cin.getline(teksts, MAX\_GARUMS);

    char\* saisinatsTeksts = saisinatTekstu(teksts);

    cout << "Saisināta teksta rindiņa: " << saisinatsTeksts << endl;

    delete[] saisinatsTeksts; // Atbrīvo dinamiski rezervēto atmiņu

    return 0;

}

Uzrakstīt C++ funkciju, kas dotajā zema līmeņa simbolu virknē atrod visas vietas, kur parādās skaitlis 150 un izdzēš visus šos skaitļus no virknes.

void delete150(char \* s) {

    if (strlen(s)<3) return;

    for (int i=0;s[i+2]!='\0';i++)

        if (s[i]=='1' && s[i+1]=='5' && s[i+2]=='0')

            for (int j=i--;s[j+2]!='\0';j++)

                s[j]=s[j+3];

}

///Izveidot C++ funkciju, kas ļauj noskaidrot, cik reizes augsta līmeņa simbolu virknē ir sastopams konkrēts simbols.

#include <iostream>

#include <string>

using namespace std;

int skaititSimbolu(const string& virkne, char mekletaisSimbols) {

    int skaititajs = 0;

    for (char simbols : virkne) {

        if (simbols == mekletaisSimbols) {

            skaititajs++;

        }

    }

    return skaititajs;

}

int main() {

    string teksts;

    char simbols;

    cout << "Ievadiet tekstu: ";

    getline(cin, teksts);

    cout << "Ievadiet meklējamo simbolu: ";

    cin >> simbols;

    int skaits = skaititSimbolu(teksts, simbols);

    cout << "Simbols '" << simbols << "' parādās " << skaits << " reizes." << endl;

    return 0;

}

///Sastādīt C++ klasi ”Punkts” izmantojot objektorientētās programmēšanas līdzekļus

// konstruktors; destruktors;

// setPunkts(x, y) – novieto objektu punktā ar koordinātēm x un y;

// getPunkts(x, y) – atgriež objekta koordinātes parametros x un y;

// increasePunkts(d) – palielina katras kooordinātes vērtību par doto vērtību d;

// print() – izdrukā objekta koordinātes.

// Punkts.h

class Punkts {

public:

    Punkts(); // Konstruktors

    ~Punkts(); // Destruktors

    void setPunkts(int x, int y);

    void getPunkts(int &x, int &y) const;

    void increasePunkts(int d);

    void print() const;

private:

    int x, y;

};

// Punkts.cpp

#include "Punkts.h"

#include <iostream>

using namespace std;

Punkts::Punkts() : x(0), y(0) {}

Punkts::~Punkts() {}

void Punkts::setPunkts(int x, int y) {

    this->x = x;

    this->y = y;

}

void Punkts::getPunkts(int &x, int &y) const {

    x = this->x;

    y = this->y;

}

void Punkts::increasePunkts(int d) {

    x += d;

    y += d;

}

void Punkts::print() const {

    cout << "Punkts(" << x << ", " << y << ")" << endl;

}

// main.cpp

#include "Punkts.h"

#include <iostream>

using namespace std;

int main() {

    // Izveido objektu tiešā veidā

    Punkts p1;

    p1.setPunkts(5, 10);

    p1.print();

    // Izveido objektu dinamiskā veidā

    Punkts \*p2 = new Punkts();

    p2->setPunkts(15, 20);

    p2->print();

    p1.increasePunkts(3);

    p1.print();

    p2->increasePunkts(3);

    p2->print();

    int x, y;

    p1.getPunkts(x, y);

    cout << "P1 koordinātes: (" << x << ", " << y << ")" << endl;

    p2->getPunkts(x, y);

    cout << "P2 koordinātes: (" << x << ", " << y << ")" << endl;

    delete p2;

    return 0;

}

Izveidot C++ programmu, kas izveido saīsinātu teksta rindiņu, ņemot tikai katru otro simbolu, sākot ar rindiņas pirmo simbolu. Tātad no “123 56” ir jāiegūst “135”.

Teksta rindiņu ievada lietotājs. Gan dotā teksta rindiņa, gan izveidotā teksta rindiņa jāsaglabā programmā kā zema līmeņa simbolu virknes.

Dotās teksta rindiņas apstrādei sastādīt funkciju, kura no saņemtās rindiņas izveido jaunu rindiņu un atgriež norādi uz jauno rindiņu.

#include <iostream>

#include <cstring>

using namespace std;

char\* saisinatTekstu(const char\* ievadTeksts) {

    int garums = strlen(ievadTeksts);

    char\* saisinats = new char[garums / 2 + 1]; // +1 for null terminator

    for (int i = 0, j = 0; i < garums; i += 2, j++) {

        saisinats[j] = ievadTeksts[i];

    }

    saisinats[garums / 2] = '\0'; // Pievieno beigu simbolu

    return saisinats;

}

int main() {

    const int MAX\_GARUMS = 1000;

    char teksts[MAX\_GARUMS];

    cout << "Ievadiet tekstu: ";

    cin.getline(teksts, MAX\_GARUMS);

    char\* saisinatsTeksts = saisinatTekstu(teksts);

    cout << "Saisināta teksta rindiņa: " << saisinatsTeksts << endl;

    delete[] saisinatsTeksts; // Atbrīvo dinamiski rezervēto atmiņu

    return 0;

}

Uzrakstīt C++ funkciju, kas dotajā zema līmeņa simbolu virknē atrod visas vietas, kur parādās skaitlis 150 un izdzēš visus šos skaitļus no virknes.

void delete150(char \* s) {

    if (strlen(s)<3) return;

    for (int i=0;s[i+2]!='\0';i++)

        if (s[i]=='1' && s[i+1]=='5' && s[i+2]=='0')

            for (int j=i--;s[j+2]!='\0';j++)

                s[j]=s[j+3];

}