Saman Jaafar (GT90SM)

**CPP Optional homework**

1. **Hello World**

#include <iostream>

#include <cstdio>

//using namespace std;

int main() {

std::cout<<"Hello, World!";

return 0;

}

1. **Input and Output**

#include <iostream>

using namespace std;

int main() {

int a, b, c;

cin >> a >> b >> c;

cout << a + b + c << endl;

}

# Basic Data Types

#include <iostream>

#include <cstdio>

using namespace std;

int main() {

int a;

long b;

char c;

float d;

double e;

cin >> a >> b >> c >> d >> e;

cout << a << endl;

cout << b << endl;

cout << c << endl;

cout.precision(3);

cout << fixed << d << endl;

cout.precision(9);

cout << fixed << e << endl;

return 0;

}

# Conditional Statements

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin >> n;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

if (1 <= n <= 9)

{

switch (n){

case 1:cout << "one";

break;

case 2: cout << "two";

break;

case 3: cout << "three";

break;

case 4: cout << "four";

break;

case 5: cout << "five";

break;

case 6: cout << "six";

break;

case 7: cout << "seven";

break;

case 8: cout << "eight";

break;

case 9: cout << "nine";

break;

}

}

if(n>=10) {cout<<"Greater than 9";}

return 0;

}

# For Loop

#include <iostream>

#include <cstdio>

using namespace std;

int main() {

int a,b ;

cin>>a;

cin>>b;

for (int n = a; n <= b; n++) {

if(1<= n<=9){

if (n == 1) {

cout << "one"<<endl;

} else if (n == 2) {

cout << "two" << endl;

} else if (n == 3) {

cout << "three" << endl;

} else if (n == 4) {

cout << "four" << endl;

} else if (n == 5) {

cout << "five" << endl;

} else if (n == 6) {

cout << "six" << endl;

} else if (n == 7) {

cout << "seven" << endl;

} else if (n == 8) {

cout << "eight" << endl;

} else if (n == 9) {

cout << "nine" << endl;

}}

if (n>9)

{

if (n % 2 == 0) {

cout << "even" << endl;

} else{

cout << "odd" << endl;

}

}}

return 0;

}

# Functions

#include <iostream>

#include <cstdio>

using namespace std;

int max\_of\_four(int a, int b, int c, int d) {

int max;

max = (a > b) ? a : b;

max = (max> c) ? max : c;

max = (max> d) ? max : d;

return max;

}

int main() {

int a, b, c, d;

scanf("%d %d %d %d", &a, &b, &c, &d);

int ans = max\_of\_four(a, b, c, d);

printf("%d", ans);

return 0;

}

# Arrays Introduction

#include <iostream>

int main() {

int N, i = 0;

std::cin >> N;

int \*A = new int[N];

while (std::cin >> A[i++]);

while (std::cout << A[--N] << ' ' && N);

delete[] A;

return 0;

}

# Variable Sized Arrays

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

// get length of array 'a' and number of queries

int n, q;

cin >> n >> q;

// create vector of vectors

vector<vector<int>> a(n);

// fill each 2D vector i with k\_i values

for (int i = 0; i < n; i++) {

// get the length k of the vector at a[i]

int k;

cin >> k;

// fill the vector with k values

a[i].resize(k);

for (int j = 0; j < k; j++) {

cin >> a[i][j];

}

}

// run queries on a

for (int q\_num = 0; q\_num < q; q\_num++) {

// get i, j as the 'query' to get a value from a

int i, j;

cin >> i >> j;

cout << a[i][j] << endl;

}

return 0;}

# Attribute Parser

#include <iostream>

#include <vector>

enum node\_type { tag, label, value };

struct parse\_tree\_node {

node\_type nt;

std::string name;

std::vector<struct parse\_tree\_node \*> children;

struct parse\_tree\_node \*parent;

};

void utility\_free(struct parse\_tree\_node \*ptn) {

for (auto const &value : ptn->children)

utility\_free(value);

delete ptn;

}

int main(void) {

struct parse\_tree\_node \*root = new struct parse\_tree\_node;

struct parse\_tree\_node \*current\_level = root;

current\_level->name = "ROOT";

current\_level->nt = tag;

int n, q;

std::cin >> n >> q;

// parse lines into tree

for (int i = 0; i < n; i++) {

while (std::cin.get() != '<');

if (std::cin.peek() != '/') {

char next\_char;

bool expecting\_space = true;

bool need\_next = false;

bool done = false;

bool quoted\_mode = false;

struct parse\_tree\_node \*new\_node = new struct parse\_tree\_node;

new\_node->nt = tag;

current\_level->children.push\_back(new\_node);

new\_node->parent = current\_level;

current\_level = new\_node;

while (!done) {

next\_char = std::cin.get();

if (!quoted\_mode)

switch (next\_char) {

case '"':quoted\_mode = true;need\_next = true;break;

case '>':done = true;break;

case ' ':

if (!expecting\_space)

need\_next = true;break;

case '=': {

struct parse\_tree\_node \*parent\_node = new\_node;

new\_node = new struct parse\_tree\_node;

new\_node->nt = value;

parent\_node->children.push\_back(new\_node);

expecting\_space = true;

need\_next = false;break;

}

default:

if (need\_next) {

struct parse\_tree\_node \*parent\_node = current\_level;

;

new\_node = new struct parse\_tree\_node;

new\_node->nt = label;

parent\_node->children.push\_back(new\_node);

need\_next = false;

}

new\_node->name.append(1, next\_char);

expecting\_space = false;

break;

}

else {

if (next\_char != '"')

new\_node->name.append(1, next\_char);

else

quoted\_mode = false;

}

}

} else {

current\_level = current\_level->parent;

while (std::cin.get() != '>')

;

}

}

// process queries

for (int i = 0; i < q; i++) {

struct parse\_tree\_node \*current\_level = root;

std::string qstr\_part;

char peek\_char;

bool tilde\_mode = false;

while ((peek\_char = std::cin.peek()) == ' ' || peek\_char == '\n')

std::cin.get();

while (true) {

char next\_char = std::cin.get();

if (next\_char == '.' || next\_char == '~') {

bool found = false;

tilde\_mode = next\_char == '~';

for (auto const &value : current\_level->children)

if (value->nt == tag && value->name == qstr\_part) {

current\_level = value;

qstr\_part = "";

found = true;

break;

}

if (!found) {

char drop\_char;

std::cout << "Not Found!" << std::endl;

while ((drop\_char = std::cin.get()) != '\n' && drop\_char != EOF)

;

break;

}

} else if (next\_char == '\n' || next\_char == EOF) {

bool found = false;

if (tilde\_mode)

for (auto const &value : current\_level->children)

if (value->nt == label && value->name == qstr\_part) {

// This next line is a little sketchy.

std::cout << value->children[0]->name << std::endl;

found = true;

break;

}

if (!found)

std::cout << "Not Found!" << std::endl;

break;

} else {

qstr\_part.append(1, next\_char);

}

}

}

utility\_free(root);

return 0;

}

# StringStream

#include <sstream>

#include <vector>

#include <iostream>

using namespace std;

vector<int> parseInts(string str) {

    // Complete this function

stringstream ss(str);

int a;

vector<int> vec;

while (ss >> a) {

vec.push\_back(a);

if (ss.peek() == ',') {

ss.ignore();

}

}

return vec;

}

int main() {

string str;

cin >> str;

vector<int> integers = parseInts(str);

for(int i = 0; i < integers.size(); i++) {

cout << integers[i] << "\n";

}

return 0;

}

# Strings

#include <iostream>

#include <string>

using namespace std;

int main() {

char change;

string a;

string b;

cin>>a>>b;

cout << a.length()<<" "<< b.length()<<endl;

string c = a + b; // c = "abcdef".

cout <<c<<endl;

change = a[0];

a[0]=b[0];

b[0]=change;

cout<<a<<' '<<b;

return 0;

}

# Structs

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

/\*

add code for struct here.

\*/

struct Student {

int age;

string first\_name;

string last\_name;

int standard;

};

int main() {

Student st;

cin >> st.age >> st.first\_name >> st.last\_name >> st.standard;

cout << st.age << " " << st.first\_name << " " << st.last\_name << " " << st.standard;

return 0;

}

# Class

#include <iostream>

#include <sstream>

using namespace std;

/\*

Enter code for class Student here.

Read statement for specification.

\*/

class Student {

public:

Student() {

m\_age = 0;

m\_first = "";

m\_last = "";

m\_stan = 0;

}

int get\_age() { return m\_age; }

void set\_age(int n\_age) { m\_age = n\_age; }

string get\_first\_name() { return m\_first; }

void set\_first\_name(string name) { m\_first = name; }

string get\_last\_name() { return m\_last; }

void set\_last\_name(string name) { m\_last = name; }

int get\_standard() { return m\_stan; }

void set\_standard(int standard) { m\_stan = standard; }

string to\_string() {

string final = std::to\_string(m\_age) + "," + m\_first + "," + m\_last + "," +

std::to\_string(m\_stan);

return final;

}

private:

int m\_age;

string m\_first;

string m\_last;

int m\_stan;

};

int main() {

Student st;

int age, standard;

string first\_name, last\_name;

cin >> age >> first\_name >> last\_name >> standard;

st.set\_age(age);

st.set\_standard(standard);

st.set\_first\_name(first\_name);

st.set\_last\_name(last\_name);

cout << st.get\_age() << "\n";

cout << st.get\_last\_name() << ", " << st.get\_first\_name() << "\n";

cout << st.get\_standard() << "\n";

cout << "\n";

cout << st.to\_string();

return 0;

}

# Classes and Objects

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

#include <cassert>

using namespace std;

// Write your Student class here

class Student {// class of Student

private:

int scores[5]; //the scores of 5 exam of students

int sum;// sumension of any student scores

public:

Student() : sum(0) {}

int calculateTotalScore() { return sum; }// total score of any student

void input() {// input the scores of each student

for (int i = 0; i < 5; i++) {

cin >> scores[i]; // input 5 scores for each student.

sum += scores[i];// the sum of the all 5th scores.

}

}

};

int main() {

int n; // number of students

cin >> n;

Student \*s = new Student[n]; // an array of n students

for(int i = 0; i < n; i++){

s[i].input();

}

// calculate kristen's score

int kristen\_score = s[0].calculateTotalScore();

// determine how many students scored higher than kristen

int count = 0;

for(int i = 1; i < n; i++){

int total = s[i].calculateTotalScore();

if(total > kristen\_score){

count++;

}

}

// print result

cout << count;

return 0;

}

# Box It!

#include<bits/stdc++.h>

using namespace std;

//Implement the class Box

class Box

{private:

int l, b, h; // l,b,h are integers representing the dimensions of the box

// The class should have the following functions :

// Constructors:

// Box();

public:

Box() {

l = 0;

b = 0;

h = 0;

}

// Box(int,int,int);

Box(int length, int breadth, int height){

l = length;

b = breadth;

h = height;}

// Box(Box);

Box(const Box& B){

l = B.l;

b = B.b;

h = B.h;}

int getLength(){ // Return box's length

return l;}

int getBreadth() { // Return box's breadth

return b;}

int getHeight (){ //Return box's height

return h;

}

long long CalculateVolume(){ // Return the volume of the box

return (long long) l\*b\*h;}

// Overload operator < as specified

friend bool operator<(Box &A, Box &B) {

if ((A.l < B.l) || ((A.b < B.b) && (A.l == B.l)) ||

((A.h < B.h) && (A.l == B.l) && (A.b == B.b))) {

return true;

}else{

return false;

}

};

//ostream& operator<<(ostream& out, Box& B)

friend ostream& operator<<(ostream& out, Box& B){

cout << B.l << " " << B.b << " " << B.h;

return cout;

}

};

void check2()

{

    int n;

    cin>>n;

    Box temp;

    for(int i=0;i<n;i++)

    {

        int type;

        cin>>type;

        if(type ==1)

        {

            cout<<temp<<endl;

        }

        if(type == 2)

        {

            int l,b,h;

            cin>>l>>b>>h;

            Box NewBox(l,b,h);

            temp=NewBox;

            cout<<temp<<endl;

        }

        if(type==3)

        {

            int l,b,h;

            cin>>l>>b>>h;

            Box NewBox(l,b,h);

            if(NewBox<temp)

            {

                cout<<"Lesser\n";

            }

            else

            {

                cout<<"Greater\n";

            }

        }

        if(type==4)

        {

            cout<<temp.CalculateVolume()<<endl;

        }

        if(type==5)

        {

            Box NewBox(temp);

            cout<<NewBox<<endl;

        }

    }

}

int main()

{

    check2();

}