6.11

输出：

7.5

7

0

0

6.4

-6

-14

6.12

*#include* <iostream>

*#include* <iomanip>

using namespace std;

double calculateCharges(double *hours*) {

*if* (*hours* <= 3.0) {

*return* 2.00;

    } *else* *if* (*hours* <= 24.0) {

        double charge = 2.00 + 0.50 \* (*hours* - 3.0);

*return* (charge > 10.00) ? 10.00 : charge;

    }

*return* 10.00; *// Maximum charge for 24 hours*

}

int main() {

    double hours1, hours2, hours3;

    cout << "Enter hours parked for car 1: ";

    cin >> hours1;

    cout << "Enter hours parked for car 2: ";

    cin >> hours2;

    cout << "Enter hours parked for car 3: ";

    cin >> hours3;

    double charge1 = calculateCharges(hours1);

    double charge2 = calculateCharges(hours2);

    double charge3 = calculateCharges(hours3);

    double totalHours = hours1 + hours2 + hours3;

    double totalCharges = charge1 + charge2 + charge3;

    cout << fixed << setprecision(2);

    cout << "Car\tHours\tCharge\n";

    cout << "1\t" << hours1 << "\t" << charge1 << "\n";

    cout << "2\t" << hours2 << "\t" << charge2 << "\n";

    cout << "3\t" << hours3 << "\t" << charge3 << "\n";

    cout << "TOTAL\t" << totalHours << "\t" << totalCharges << "\n";

*return* 0;

}

6.26

*#include* <iostream>

*#include* <iomanip>

using namespace std;

*// Function to convert Fahrenheit to Celsius*

double celsius(int *fahrenheit*) {

*return* (*fahrenheit* - 32) \* 5.0 / 9.0;

}

*// Function to convert Celsius to Fahrenheit*

double fahrenheit(int *celsius*) {

*return* *celsius* \* 9.0 / 5.0 + 32;

}

int main() {

    cout << fixed << setprecision(2);

*// Print Celsius to Fahrenheit chart*

    cout << "Celsius to Fahrenheit conversion chart:\n";

    cout << "Celsius\tFahrenheit\n";

*for* (int c = 0; c <= 100; ++c) {

        cout << c << "\t" << fahrenheit(c) << "\n";

    }

    cout << "\n";

*// Print Fahrenheit to Celsius chart*

    cout << "Fahrenheit to Celsius conversion chart:\n";

    cout << "Fahrenheit\tCelsius\n";

*for* (int f = 32; f <= 212; ++f) {

        cout << f << "\t\t" << celsius(f) << "\n";

    }

*return* 0;

}

6.27

*#include* <iostream>

using namespace std;

*// Function to find the smallest of three double-precision floating-point numbers*

double findSmallest(double *num1*, double *num2*, double *num3*) {

    double smallest = *num1*;

*if* (*num2* < smallest) {

        smallest = *num2*;

    }

*if* (*num3* < smallest) {

        smallest = *num3*;

    }

*return* smallest;

}

int main() {

    double num1, num2, num3;

    cout << "Enter three double-precision floating-point numbers: ";

    cin >> num1 >> num2 >> num3;

    double smallest = findSmallest(num1, num2, num3);

    cout << "The smallest number is: " << smallest << endl;

*return* 0;

}

6.28

*#include* <iostream>

*#include* <vector>

bool isPerfect(int *number*) {

    int sum = 0;

    std::vector<int> divisors;

*for* (int i = 1; i <= *number* / 2; ++i) {

*if* (*number* % i == 0) {

            sum += i;

            divisors.push\_back(i);

        }

    }

*if* (sum == *number*) {

        std::cout << *number* << " is a perfect number. Divisors: ";

*for* (int divisor : divisors) {

            std::cout << divisor << " ";

        }

        std::cout << std::endl;

*return* true;

    }

*return* false;

}

int main() {

    std::cout << "Perfect numbers between 1 and 1000:" << std::endl;

*for* (int i = 1; i <= 1000; ++i) {

        isPerfect(i);

    }

*// Uncomment the following lines to test numbers much larger than 1000*

*// std::cout << "Testing larger numbers:" << std::endl;*

*// for (int i = 1001; i <= 10000; ++i) {*

*//     isPerfect(i);*

*// }*

*return* 0;

}

6.29

*#include* <bits/stdc++.h>

bool isPrime(int *num*)

{

*if* (*num* == 1) *return* 0;

*if* (*num* == 2 || *num* == 3) *return* 1;

*if* (*num* % 6 != 1 && *num* % 6 != 5) *return* 0;

    int tmp = sqrt(*num*);

*for* (int i = 5; i <= tmp; i += 6)

*if* (*num* % i == 0 || *num* % (i + 2) == 0) *return* 0;

*return* 1;

}

6.30

*#include* <iostream>

using namespace std;

*// Function to reverse the digits of an integer*

int reverseDigits(int *num*) {

    int reversed = 0;

*while* (*num* != 0) {

        int digit = *num* % 10;

        reversed = reversed \* 10 + digit;

*num* /= 10;

    }

*return* reversed;

}

int main() {

    int number;

    cout << "Enter an integer: ";

    cin >> number;

    int reversedNumber = reverseDigits(number);

    cout << "The number with its digits reversed is: " << reversedNumber << endl;

*return* 0;

}

6.37

*#include* <iostream>

*#include* <limits>

using namespace std;

*// Function to calculate the nth Fibonacci number using int*

int fibonacciInt(int *n*) {

*if* (*n* <= 1) *return* *n*;

    int a = 0, b = 1, c;

*for* (int i = 2; i <= *n*; ++i) {

        c = a + b;

        a = b;

        b = c;

    }

*return* b;

}

*// Function to calculate the nth Fibonacci number using double*

double fibonacciDouble(int *n*) {

*if* (*n* <= 1) *return* *n*;

    double a = 0, b = 1, c;

*for* (int i = 2; i <= *n*; ++i) {

        c = a + b;

        a = b;

        b = c;

    }

*return* b;

}

int main() {

*// Part (a): Using int*

    cout << "Fibonacci numbers using int:\n";

    int n = 0;

*while* (true) {

        int fib = fibonacciInt(n);

*if* (fib < 0) *break*; *// Overflow detected*

        cout << "Fibonacci(" << n << ") = " << fib << endl;

        ++n;

    }

    cout << "The largest int Fibonacci number is Fibonacci(" << n-1 << ") = " << fibonacciInt(n-1) << endl;

*// Part (b): Using double*

    cout << "\nFibonacci numbers using double:\n";

    n = 0;

*while* (true) {

        double fib = fibonacciDouble(n);

*if* (fib == numeric\_limits<double>::infinity()) *break*; *// Overflow detected*

        cout << "Fibonacci(" << n << ") = " << fib << endl;

        ++n;

    }

    cout << "The largest double Fibonacci number is Fibonacci(" << n-1 << ") = " << fibonacciDouble(n-1) << endl;

*return* 0;

}

6.39

*#include* <bits/stdc++.h>

using namespace std;

void move\_iterate(int *n*, char *des*, char *now*, char *temp*)

{

    char inorder[2][4] = {'0', *temp*, *des*, *now*, '0', *des*, *now*, *temp*};

    int i, j, k;

    char\* np = new char[*n* + 1];

*for* (int i = 0; i < *n* + 1; i++) np[i] = 'A';

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

*for* (j = i, k = 1; j % 2 == 0; k++, j /= 2);

        cout << "将圆盘" << k << "从 " << np[k] << " 移动到 ";

        np[k] = inorder[(1 + *n* + k) % 2][np[k] - 'A' + 1];

        cout << np[k] << endl;

    }

}

int main()

{

    int n;

*while* (true) {

        cout << "请输入汉诺塔高度:";

        cin >> n;

        cout << "默认从A移动到C" << endl;

        move\_iterate(n, 'C', 'A', 'B');

    }

}

6.40

*#include* <iostream>

using namespace std;

int main() {

    static int count = 1;

    cout << "main called " << count++ << " times" << endl;

*// Limit the recursion to prevent stack overflow*

*if* (count <= 10) {

        main();

    }

*return* 0;

}