Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение   
высшего образования   
«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»  
(ВлГУ)

Институт прикладной математики, физики и информатики

Кафедра физики и прикладной математики

ЛАБОРАТОРНАЯ РАБОТА №3  
по дисциплине  
«Основы программирования»  
на тему:  
«Перегрузка функций»

Выполнил:  
ст. гр. ПМИ-123

Рушев А.М.  
  
  
Принял:  
ст. преподаватель   
каф. ФиПМ  
Шишкина М.В.

Владимир,2024

**Цель работы**

Изучение механизма перегрузки функций на языке *С++*. Закрепление навыков разработки функций на языке *С++*, способов передачи параметров в функции, возвращения значения из функции, использование этого значения в точке вызова.

**Постановка задачи**

1. Написать функцию вычисления суммы элементов двумерного целочисленного массива и функцию сортировки двумерного целочисленного массива.
2. Перегрузить, описанные выше функции, для работы с массивами символов. При этом, функция сложения должна возвращать предложение. Т.е. строку, образованную сложением строк массива, слиянием их через пробел с добавлением точки после последнего символа. Функция сортировки строк, должна упорядочить строки (не символы) исходного массива по алфавиту.
3. Объявить указатель на функцию, настроить его на описанную выше функцию вычисления суммы элементов двумерного массива. Вызвать соответствующую функцию через этот указатель.

**Теоретическая часть**

Язык С++ позволяет определять функции с одним и тем же именем, но разным набором параметров. Подобная возможность называется перегрузкой функций. Компилятор же на этапе компиляции на основании параметров выберет нужный тип функции.

Чтобы определить несколько различных версий функции с одним и тем же именем, все эти версии должны отличаться как минимум по одному из следующих признаков:

* имеют разное количество параметров
* соответствующие параметры имеют разный тип

При этом различные версии функции могут также отличаться по возвращаемому типу. Однако компилятор, когда выбирает, какую версию функции использовать, ориентируется именно на количество параметров и их тип.

Пример:

int sum(int a, int b)

{

return a + b;

}

double sum(double a, double b)

{

return a + b;

}

int main()

{

int result1 {sum(3, 6)};

std::cout << result1 << std::endl; //9

double result2 {sum(3.3, 6.6)};

std::cout << result2 << std::endl; // 9.9

}

Здесь определены две версии функция sum, которая складывает два числа. В одном случае она складывает два числа типа int, в другом – числа типа double. При вызове функций компилятор на основании переданных аргументов определяет, какую версию использовать.

При перегрузке функций с параметрами-ссылками следует учитывать, что параметры типов и ссылок на такие же типы не различаются при перегрузке. Например, два следующих прототипа:

void print(int);

void print(int&);

Не считаются разными версиями функции print.

При перегрузке функций константный параметр отличается от неконстантного параметра только для ссылок и указателей. В остальных случаях константый параметр будет идентичен неконстантному параметру. Например, следующие два прототипа при перегрузке различаться НЕ будут:

void print(int);

void print(const int);

Во втором прототипе компилятор игнорирует оператор const.

**Практическая часть**

**Листинг кода программы:**

#include <iostream>

#include <string>

using namespace std;

const int rows = 4;

const int \_lenght = 100;

const int n = 2;

const int m = 6;

//Вывод массива

void PrintMassiv(int\*\* mass, int n, int m) {

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

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

cout << mass[i][j] << "\t";

}

cout << endl;

}

cout << endl;

}

//Сумма элементов массива целых чисел

int SumMassiv(int\*\* mass, int n, int m) {

int sum = 0;

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

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

sum += mass[i][j];

}

}

return sum;

}

//Сортировка массива целых чисел

void SortMassiv(int\*\* mass, int n, int m) {

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

{

for (int j = 0; j < m - 1; j++)

{

for (int k = 0; k < m - j - 1; k++) {

if (mass[i][k] > mass[i][k + 1])

{

int temp = mass[i][k];

mass[i][k] = mass[i][k + 1];

mass[i][k + 1] = temp;

}

}

}

}

}

//Сложение массивов символов

char\* SumMassiv(char charMassiv[][6], int n, int m) {

int k = 0;

char\* resMassiv = new char[n \* m + 1 + n];

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

for (int j = 0; j < m - 1; j++) {

resMassiv[k++] = charMassiv[i][j];

if (i == n - 1 && j == m - 2)

{

resMassiv[k++] = '.';

}

}

resMassiv[k++] = ' ';

}

resMassiv[k] = '\0';

resMassiv[0] = toupper(resMassiv[0]);

return resMassiv;

}

void SortMassiv(char charMassiv[][6], int n, int m) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (strcmp(charMassiv[j], charMassiv[j + 1]) > 0) {

char temp[m + 1];

strcpy\_s(temp, charMassiv[j]);

strcpy\_s(charMassiv[j], charMassiv[j + 1]);

strcpy\_s(charMassiv[j + 1], temp);

}

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Задание 1." << endl;

//создание массива целых чисел

int\*\* mass = new int\* [m];

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

mass[i] = new int[m];

}

//Заполнение массива целых чисел

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

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

mass[i][j] = 1 + rand() % 50;

}

}

//вывод массива целых чисел

PrintMassiv(mass, n, m);

int sumTask1 = SumMassiv(mass, n, m);

cout << "\nСумма элементов массива = " << sumTask1 << endl;

//сортировка целочисленного массива

SortMassiv(mass, n, m);

cout << "\nОтсортированный массив:" << endl;

//вывод целочисленного массива

PrintMassiv(mass, n, m);

cout << "\nЗадание 2." << endl;

char \_charMass[n][m] = { {"world"}, {"hello"} };

const char\* resMass = new char[n \* m + 1];

//сложение массива символов

resMass = SumMassiv(\_charMass, n, m);

cout << "Сумма массива символов: " << resMass << endl;

//сортировка массива символов

SortMassiv(\_charMass, n, m);

//вывод массива символов

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

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

cout << \_charMass[i][j];

}

cout << endl;

}

//вывод сложения массива символов отсортированных

resMass = SumMassiv(\_charMass, n, m);

cout << "Сумма массива символов: " << resMass << endl;

cout << "\nЗадание 3." << endl;

char\* (\*ptr)(char[][m], int, int) = &SumMassiv;

cout << "Сумма символьного массива: " << (\*ptr)(\_charMass, n, m) << endl;

int(\*pf2)(int\*\*, int, int) = &SumMassiv;

int resultTask3 = (\*pf2)(mass, n, m);

cout << "Сумма целочисленного массива: " << resultTask3 << endl;

//Освобождение памяти у массива

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

delete[] mass[i];

}

delete[] mass;

system("pause");

return 0;

}

**Результат выполнения кода программы:**

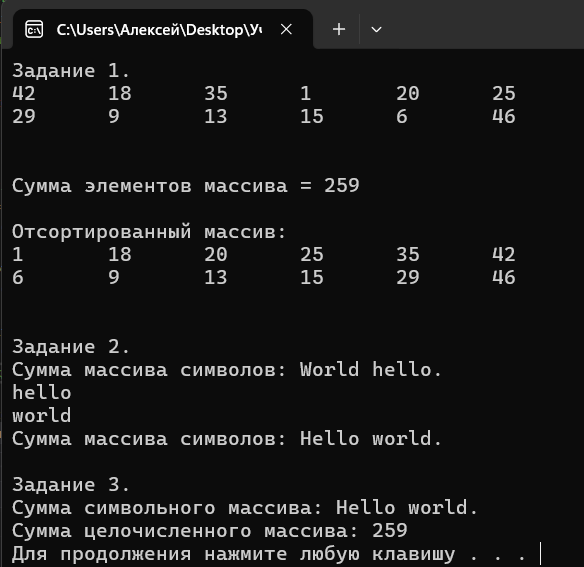


Рисунок 1 – Результат выполнения кода программы

**Вывод:** в ходе лабораторной работы были изучены и применены методы работы с перегрузкой функций на языке C++. Перегрузка позволяет определять несколько функций с одним именем, но с разными параметрами. Преимуществами перегрузки функций являются удобство и гибкость использования. Она позволяет создавать более удобные интерфейсы и выполнять различные действия с разными типами или количеством параметров. Важным условием перегрузки функций является различие в типах и/или количестве параметров. Помимо выполнения разных действий, перегрузка функций также может обеспечивать безопасность кода. Например, она позволяет предотвратить ошибки при передаче некорректных аргументов. Также перегрузка позволяет вместо создания нескольких функций с разными именами для выполнения похожих задач, просто перегрузить одну функцию, что упрощает структуру программы. В результате работы получены практические навыки по перегрузке функций.