ФГБОУ ВО “Чувашский государственный университет им. И. Н. Ульянова”
Факультет: ИВТ
Кафедра: Вычислительной техники
Предмет: Объектно-ориентированное программирование

Лабораторная работа №5

Функции. Перегрузка функций. Шаблоны функций

Выполнил: студент группы ИВТ-41-20
Галкин Дмитрий
Проверил: доцент Обломов И.А.

Чебоксары 2021 г.

Теория

Функция – это именованная последовательность описаний и операторов, выполняющая законченное действие. Функции в C++ понимается как простейший способ модульности программы.

Функция перед ее вызовом, должна быть объявлена и определена. Объявление функции (прототип, заголовок) содержит имя функции, тип возвращаемого результата и список параметров.

Общий формат объявления функции:

```
[класс] тип_результата   имя_функции  ([список параметров])  [throw] {
        // тело функции – последовательность
        // описаний и операторов
}
```

Модификаторы доступа:

- *extern* – глобальная видимость во всех модулях проекта (по умолчанию)
- *static* – видимость только в пределах данного модуля

Индивидуальное задание

1. *Передача в функцию параметров стандартных типов*. Написать функцию вывода таблицы значения функции из лабораторной работы №2 для аргументов, изменяющихся в заданных пределах с заданным шагом, с точностью ε. Значение аргумента и точность передать в качестве параметров функции.

```
inline void showTable(double STX, double ENX, double dx, double e) {
   cout << '|' << setw(3) << 'x' << setw(3) << '|' << setw(12) << "arctg(x)"
      << setw(5) << '|' << setw(12) << "atan(x)" << setw(5) << '|' << setw(3) << 'n' << setw(3) << '|' << endl;

double STX, ENX;
double dx;
double e = 0.5;

STX = 1.5;
ENX = 7.5;
dx = 0.5;
e = 0.000001;

showTable(STX, ENX, dx, e);
cout << endl;
```

2. *Передача в функцию указателя на функцию*. Пользуясь функцией из задания №1, объявить указатель на нее и передать его как параметр некоторой другой функции.

```
// pointer func
typedef void(*ptr_showTable)(double, double, double, double);
static void pointerShowTable(ptr_showTable pst, double STX, double ENX, double dx, double e);

static void pointerShowTable(ptr_showTable pst, double STX, double ENX, double dx, double e) {
   pst(STX, ENX, dx, e);
}

pointerShowTable(showTable, STX, ENX, dx, e);
cout<<endl;
```

3. *Передача одномерных массивов в функцию*. Пользуясь массивом, определенным в пункте А лабораторной работы №3, определить функции, реализующие подпункты данного пункта.

```
//Exercise 3
int max(int *arr, const int size) { // Paragraph 1
    int max = -1000;
    for (int i = 0; i < size; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
    }

    return max;
}

long int multiplication(int *arr, int index_1, int index_2) {
    long int multiplication = 1;
    for (int i = index_1 + 1; i < index_2; i++) {
        multiplication *= arr[i];
    }

    return multiplication;
}

void transformation(int *arr_1, int *arr_2, const int size) {
    int count = 0;
    for (int i = 1; i < size; i += 2) {
        arr_2[count] = arr_1[i];
        count++;
    }
    for (int i = 0; i < size; i += 2) {
        arr_2[count] = arr_1[i];
        count++;
    }
}

int n = 8;
    int countNegative = 0;
    bool flag = false;
    int index_1 = 0, index_2 = 0;

    int *arr_1 = new int[n] {1, 5, -4, 8, 10, -2, 7, 15};
    for (int i = 0; i < n; i++) {
        if (arr_1[i] < 0) {
            countNegative++;
        }
        if(arr_1[i] < 0 && flag == true) {
            index_2 = i;
            flag = false;
        }
        if(arr_1[i] < 0 && flag == false && countNegative == 1){
            index_1 = i;
            flag = true;
        }
    }

    cout << "Source array: "; showArray(arr_1, n);

    // Paragraph 1
```

```
    std::cout << "Maximum array element: " << max(arr_1, n) << std::endl;

    // Paragraph 2
    if(countNegative >= 2) {
        std::cout << "Multiplication of array elements: " << multiplication(arr_1, index_1, index_2) << std::endl;
    } else {
        std::cout << "There are no negative elements in the array" << std:: endl;
    }

    // Paragraph 3
    int *arr_2 = new int[n];
    transformation(arr_1, arr_2, n);
    std::cout << "Array: "; showArray(arr_1, n);
    std::cout << "Transform array: "; showArray(arr_2, n);
    std::cout << std::endl;

    // Delete
    delete [] arr_1;
    delete [] arr_2;
}
```

4. *Передача строк в функцию*. Определить функцию, считывающую строку символов (длина строки не более 100 символов), подсчитать, сколько в каждой строке числовых символов.

```
//Exercise 4
int countNumbers(char *str) {
    int count = 0;
    char next;
    char newStr = *str;
    while(newStr != '\0') {
        next = *(str + 1);
        if((newStr > '0' && newStr < '9') && (next < '0' || next > '9')) {
            count++;
        }
        str++;
        newStr = *str;
    }

    if(*(str - 1) > '0' && *(str - 1) <'9') { count++; }

    return count;
}

{
        char string[100] {"A1B128C32D100000F05K..6"};
//        cout << "Enter the string:"; cin.get(string, 100);
        cout << "Source string: " << string << endl;
        cout << "Count numbers: " << countNumbers(string) << endl;
        cout << endl;
}
```

5. *Передача многомерных массивов в функцию*. Пользуясь массивом, определенным в пункте В лабораторной работы №3, определить функции, реализующие подпункты данного пункта.

```
//Exercise 5
void sum(int **arr, const int line, const int column) {
```

```
    long int sum = 0;
    for (int j = 0; j < column; ++j) {
        sum += arr[line][j];
    }
    std::cout << line + 1 << " line = " << sum << std:: endl;
}

const int line = 4, column = 4;
int countNegative = 0;
int **arr_1 = new int* [line];
double **arr_2 = new double* [line];
for (int i = 0; i < line; i++) {
    arr_1[i] = new int[column];
    arr_2[i] = new double[column];
}

for (int i = 0; i < line; ++i) {
    for (int j = 0; j < column; ++j) {
        if(i % 2 == 0 && j == 0) {
            countNegative++;
            arr_1[i][j] = -(rand() % 5 + 1);
            arr_2[i][j] = (double)rand()/(double) RAND_MAX*(1.0 - 5.0) + 5.0;
            continue;
        }
        arr_1[i][j] = rand() % 5 + 1;
        arr_2[i][j] = (double)rand()/(double) RAND_MAX*(1.0 - 5.0) + 5.0;
    }
}

showArray(arr_1, line, column);

// Paragraph 1
std::cout << "k*k: ";
for (int i = 0; i < line; ++i) {
    std::cout << arr_1[i][i] << ' ';
}
std::cout << std::endl;

// Paragraph 2
if(countNegative == 0) {
    std::cout << "There are no negative elements in the array" << std:: endl;
} else {
    for (int i = 0; i < line; ++i) {
        for (int j = 0; j < column; ++j) {
            if(arr_1[i][j] < 0) {
                sum(arr_1, i, column);
                break;
            }
        }
    }
}

cout << endl;
```

6. *Передача структур в функцию*. Определить функцию, получающую в качестве аргумента структуру и выводящую поля данной структуры.

```
//Exercise 6
void showStruct(struct Person p) {;
    cout << "Person {" << endl;
```

```
    cout << "\tName: " << p.name << endl;
    cout << "\tAge: " << p.age << endl;
    cout << "\tSex: " << ((p.sex == 1) ? "Man" : "Woman") << " }\n" << endl;
}

{
    Person p = {"Dima", 19, 1};
    showStruct(p);
}
```

7. *Рекурсивные функции*. Написать функцию упорядочивания массива по возрастанию, используя рекурсию.

```
//Exercise 7
void compareTo(int *arr, const int size, int l) {
    if(l != size) {
        for (int i = size - 1; i > l; --i) {
            if(arr[i] < arr[i - 1]) {
                swap(arr[i], arr[i - 1]);
            }
        }
        compareTo(arr, size, ++l);
    }
}

{
    int *arr = new int[10];
    for (int i = 0; i < 10; ++i) {
        arr[i] = rand() % 15 + 1;
    }

    cout << "Source array: "; showArray(arr, 10);
    compareTo(arr, 10, 0);
    cout << "Sort ascending: "; showArray(arr, 10);
    cout << endl;

    delete [] arr;
}
```

8. *Перегружаемые функции*. Пользуясь заданием №3 данной работы, перегрузить функцию для массивов типов int и double.

```
{
    int arrInt[4] {1, 2, 3, 4};
    double arrDouble[4] {0.1, 0.5, 8.4, 3.1};
    cout << "Array int: "; showArray(arrInt, 4);
    cout << "Array double: "; showArray(arrDouble, 4);
    cout << endl;
}
```

9. *Шаблоны функции*. Определить шаблон функции, реализующий подпункт 1 пункта В (или пункт В) лабораторной работы №3 для произвольных арифметических типов. Вызвать шаблон как обычную функцию и со спецификатором шаблона.

```
//Exercise 9
void showArray(double **arr, const int line, const int column) {
    cout << "  Array: " << endl;
    for (int i = 0; i < line; ++i) {
```

```
        for (int j = 0; j < column; ++j) {
            cout << arr[i][j] << ' ';
        }
        cout << endl;
    }
}

template<class Type> void showDiagonalDerivation(Type **arr, const int line, const int column) {
    showArray(arr, line, column);

    std::cout << "k*k: ";
    for (int i = 0; i < line; ++i) {
        std::cout << arr[i][i] << ' ';
    }
    std::cout << std::endl;
}

showDiagonalDerivation<int>(arr_1, line, column); cout << endl;
showDiagonalDerivation<double>(arr_2, line, column); cout << endl;

for (int i = 0; i < line; ++i) {
    delete [] arr_1[i];
    delete [] arr_2[i];
}
```

Текст программы:

```
#include <iostream>
#include <string>
#include <cmath>
#include <iomanip>

using namespace std;

struct Person {
    char name[50];
    int age;
    bool sex;
};
int count;


// prototype func
void showArray(int *arr, const int size);
void showArray(double arr[], const int size);
void showArray(int **arr, const int line, const int column);
void showArray(double **arr, const int line, const int column);
void showTable(double STX, double ENX, double dx, double e);
double arctg (double x, double e);
int max(int *arr, const int size);
long int multiplication(int *arr, int index_1, int index_2);
void transformation(int *arr_1, int *arr_2, const int size);
int countNumbers(char *str);
void sum(int **arr, const int line, const int column);
void showStruct(struct Person p);
void compareTo(int *arr, const int size, int);


// pointer func
```

```
typedef void(*ptr_showTable)(double, double, double, double);
static void pointerShowTable(ptr_showTable pst, double STX, double ENX, double dx, double e);

// template func
template<class Type> void showDiagonalDerivation(Type **arr, const int line, const int column);


int main() {

    std::cout << "****************** Exercise 1 ****************" << std::endl;

        double STX, ENX;
        double dx;
        double e = 0.5;

        STX = 1.5;
        ENX = 7.5;
        dx = 0.5;
        e = 0.000001;

        showTable(STX, ENX, dx, e);
        cout << endl;

    std::cout << "****************** Exercise 2 ****************" << std::endl;

        pointerShowTable(showTable, STX, ENX, dx, e);
        cout<<endl;

    std::cout << "****************** Exercise 3 ****************" << std::endl;
        {
            int n = 8;
            int countNegative = 0;
            bool flag = false;
            int index_1 = 0, index_2 = 0;

            int *arr_1 = new int[n] {1, 5, -4, 8, 10, -2, 7, 15};
            for (int i = 0; i < n; i++) {
                if (arr_1[i] < 0) {
                    countNegative++;
                }
                if(arr_1[i] < 0 && flag == true) {
                    index_2 = i;
                    flag = false;
                }
                if(arr_1[i] < 0 && flag == false && countNegative == 1){
                    index_1 = i;
                    flag = true;
                }
            }

            cout << "Source array: "; showArray(arr_1, n);

            // Paragraph 1
            std::cout << "Maximum array element: " << max(arr_1, n) << std::endl;

            // Paragraph 2
            if(countNegative >= 2) {
                std::cout << "Multiplication of array elements: " << multiplication(arr_1, index_1, index_2) <<
std::endl;
            } else {
                std::cout << "There are no negative elements in the array" << std:: endl;
```

```
        }

        // Paragraph 3
        int *arr_2 = new int[n];
        transformation(arr_1, arr_2, n);
        std::cout << "Array: "; showArray(arr_1, n);
        std::cout << "Transform array: "; showArray(arr_2, n);
        std::cout << std::endl;

        // Delete
        delete [] arr_1;
        delete [] arr_2;
    }

std::cout << "****************** Exercise 4 ****************" << std::endl;
    {
        char string[100] {"A1B128C32D100000F05K..6"};
//        cout << "Enter the string:"; cin.get(string, 100);
        cout << "Source string: " << string << endl;
        cout << "Count numbers: " << countNumbers(string) << endl;
        cout << endl;
    }

std::cout << "****************** Exercise 5 ****************" << std::endl;

        const int line = 4, column = 4;
        int countNegative = 0;
        int **arr_1 = new int* [line];
        double **arr_2 = new double* [line];
        for (int i = 0; i < line; i++) {
            arr_1[i] = new int[column];
            arr_2[i] = new double[column];
        }

        for (int i = 0; i < line; ++i) {
            for (int j = 0; j < column; ++j) {
                if(i % 2 == 0 && j == 0) {
                    countNegative++;
                    arr_1[i][j] = -(rand() % 5 + 1);
                    arr_2[i][j] = (double)rand()/(double) RAND_MAX*(1.0 - 5.0) + 5.0;
                    continue;
                }
                arr_1[i][j] = rand() % 5 + 1;
                arr_2[i][j] = (double)rand()/(double) RAND_MAX*(1.0 - 5.0) + 5.0;
            }
        }

        showArray(arr_1, line, column);

        // Paragraph 1
        std::cout << "k*k: ";
        for (int i = 0; i < line; ++i) {
            std::cout << arr_1[i][i] << ' ';
        }
        std::cout << std::endl;

        // Paragraph 2
        if(countNegative == 0) {
            std::cout << "There are no negative elements in the array" << std:: endl;
        } else {
            for (int i = 0; i < line; ++i) {
```

```
                for (int j = 0; j < column; ++j) {
                    if(arr_1[i][j] < 0) {
                        sum(arr_1, i, column);
                        break;
                    }
                }
            }
        }

        cout << endl;



    std::cout << "***************** Exercise 6 ****************" << std::endl;
        {
            Person p = {"Dima", 19, 1};
            showStruct(p);
        }

    std::cout << "***************** Exercise 7 ****************" << std::endl;
        {
            int *arr = new int[10];
            for (int i = 0; i < 10; ++i) {
                arr[i] = rand() % 15 + 1;
            }

            cout << "Source array: "; showArray(arr, 10);
            compareTo(arr, 10, 0);
            cout << "Sort ascending: "; showArray(arr, 10);
            cout << endl;

            delete [] arr;
        }

    std::cout << "***************** Exercise 8 ****************" << std::endl;
        {
            int arrInt[4] {1, 2, 3, 4};
            double arrDouble[4] {0.1, 0.5, 8.4, 3.1};
            cout << "Array int: "; showArray(arrInt, 4);
            cout << "Array double: "; showArray(arrDouble, 4);
            cout << endl;
        }

    std::cout << "***************** Exercise 9 ****************" << std::endl;

        showDiagonalDerivation<int>(arr_1, line, column); cout << endl;
        showDiagonalDerivation<double>(arr_2, line, column); cout << endl;

        for (int i = 0; i < line; ++i) {
            delete [] arr_1[i];
            delete [] arr_2[i];
        }
}


// General func
void showArray(int *arr, const int size) {
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << ' ';
    }
    cout << endl;
```

```
}

void showArray(int **arr, const int line, const int column) {
    cout << "  Array: " << endl;
    for (int i = 0; i < line; ++i) {
        for (int j = 0; j < column; ++j) {
            cout << arr[i][j] << ' ';
        }
        cout << endl;
    }
}


//Exercise 1
double arctg (double x, double e) {
    double y = M_PI / 2, a = 1;
    int n = 0;
    count = 0;
    double localX;
    do {
        localX = exp(log(x) * (2 * n + 1));
//        localX = pow(x, 2 * n + 1);
        if(count % 2 == 0) {
            a = -1 / ((2 * n + 1) * localX);
        } else {
            a = 1 / ((2 * n + 1) * localX);
        }
        y += a;
        count++;
        n++;
    } while(fabs(a) > e);
    return y;
}

inline void showTable(double STX, double ENX, double dx, double e) {
    cout << '|' << setw(3) << 'x' << setw(3) << '|' << setw(12) << "arctg(x)"
         << setw(5) << '|' << setw(12) << "atan(x)" << setw(5) << '|' << setw(3) << 'n' << setw(3) << '|' << endl;

    for(double x = STX; x <= ENX; x += dx) {
        double y = arctg(x, e);
        cout << '|' << setw(3) << x << setw(3) << '|' << setw(12) << y
             << setw(5) << '|' << setw(12) << atan(x) << setw(5) << '|' << setw(3) << count << setw(3) << '|' <<
endl;
    }
}


//Exercise 2
static void pointerShowTable(ptr_showTable pst, double STX, double ENX, double dx, double e) {
    pst(STX, ENX, dx, e);
}


//Exercise 3
int max(int *arr, const int size) { // Paragraph 1
    int max = -1000;
    for (int i = 0; i < size; i++) {
        if(arr[i] > max) {
            max = arr[i];
        }
    }
```

```
    return max;
}

long int multiplication(int *arr, int index_1, int index_2) {
    long int multiplication = 1;
    for (int i = index_1 + 1; i < index_2; i++) {
        multiplication *= arr[i];
    }

    return multiplication;
}

void transformation(int *arr_1, int *arr_2, const int size) {
    int count = 0;
    for (int i = 1; i < size; i += 2) {
        arr_2[count] = arr_1[i];
        count++;
    }
    for (int i = 0; i < size; i += 2) {
        arr_2[count] = arr_1[i];
        count++;
    }
}


//Exercise 4
int countNumbers(char *str) {
    int count = 0;
    char next;
    char newStr = *str;
    while(newStr != '\0') {
        next = *(str + 1);
        if((newStr > '0' && newStr < '9') && (next < '0' || next > '9')) {
            count++;
        }
        str++;
        newStr = *str;
    }

    if(*(str - 1) > '0' && *(str - 1) <'9') { count++; }

    return count;
}

//Exercise 5
void sum(int **arr, const int line, const int column) {
    long int sum = 0;
    for (int j = 0; j < column; ++j) {
        sum += arr[line][j];
    }
    std::cout << line + 1 << " line = " << sum << std:: endl;
}


//Exercise 6
void showStruct(struct Person p) {;
    cout << "Person {" << endl;
    cout << "\tName: " << p.name << endl;
    cout << "\tAge: " << p.age << endl;
    cout << "\tSex: " << ((p.sex == 1) ? "Man" : "Woman") << " }\n" << endl;
```

```
}


//Exercise 7
void compareTo(int *arr, const int size, int l) {
    if(l != size) {
        for (int i = size - 1; i > l; --i) {
            if(arr[i] < arr[i - 1]) {
                swap(arr[i], arr[i - 1]);
            }
        }
        compareTo(arr, size, ++l);
    }
}


//Exercise 8
void showArray(double arr[], const int size) {
    for (int i = 0; i < size; ++i) {
        cout << arr[i] << ' ';
    }
    cout << endl;
}


//Exercise 9
void showArray(double **arr, const int line, const int column) {
    cout << "  Array: " << endl;
    for (int i = 0; i < line; ++i) {
        for (int j = 0; j < column; ++j) {
            cout << arr[i][j] << ' ';
        }
        cout << endl;
    }
}

template<class Type> void showDiagonalDerivation(Type **arr, const int line, const int column) {
    showArray(arr, line, column);

    std::cout << "k*k: ";
    for (int i = 0; i < line; ++i) {
        std::cout << arr[i][i] << ' ';
    }
    std::cout << std::endl;
}
```

Вывод: Я практическим путем выяснил как использовать указатели на практики и к каким нежелательным последствиям могут привести неправильное использование указателей.