Лабораторная работа №1
Изучение функций в языке программирования С++

Цель лабораторной работы

Целью лабораторной работы является получение опыта работы с функциями в языке программирования С++.

Краткие теоретические сведения

Функция - это совокупность объявлений переменных и операторов, обычно предназначенная для решения определенной задачи. Каждая функция должна иметь имя, которое используется для ее объявления, определения и вызова. В любой программе на СИ должна быть функция с именем main (главная функция), именно с этой функции, в каком бы месте программы она не находилась, начинается выполнение программы.

При вызове функции ей при помощи аргументов (**формальных параметров**) могут быть переданы некоторые значения (фактические параметры), используемые во время выполнения функции. Функция может возвращать некоторое значение. Это возвращаемое значение и есть результат выполнения функции, который при выполнении программы подставляется в точку вызова функции, где бы этот вызов ни встретился. Допускается также использовать функции не имеющие аргументов и функции не возвращающие никаких значений.

С использованием функций в языке СИ связаны три понятия:

- ✓ определение функции (описание действий, выполняемых функцией),
- ✓ объявление функции (задание формы обращения к функции) (прототип функции) (необязательно),
- ✓ вызов функции.

Определение функции задает тип возвращаемого значения, имя функции, типы и число формальных параметров, а также объявления переменных и операторы, называемые телом функции, и определяющие действие функции. В определении функции также может быть задан класс памяти.

В соответствии с синтаксисом языка СИ определение функции имеет следующую форму:

[спецификатор-класса-памяти] [спецификатор-типа] имя-функции
([список-формальных-параметров])
{ тело-функции }

Необязательный спецификатор-класса-памяти задает класс памяти функции, который может быть *static* или *extern*. Спецификатор-типа функции задает тип возвращаемого значения и может задавать любой тип. Тип возвращаемого значения, задаваемый в определении функции, должен соответствовать типу в объявлении этой функции.

Функция возвращает значение, если ее выполнение заканчивается оператором *return*, содержащим некоторое выражение. Указанное выражение вычисляется, преобразуется, если необходимо, к типу возвращаемого значения и возвращается в точку вызова функции в качестве результата.

Различают три вида передачи переменных в функцию. Непосредственная передача, передача по указателю, передача по ссылке. При непосредственной передаче параметры функции передаются по значению и могут рассматриваться как локальные переменные, для которых выделяется память при вызове функции и производится инициализация значениями фактических параметров. При выходе из функции значения этих переменных теряются. Поскольку передача параметров происходит по значению, в теле функции нельзя изменить значения переменных в вызывающей функции, являющихся фактическими параметрами. Передача по указателю и по значению позволяет обращаться внутри функции непосредственно к переменной из программы.

Рассмотрим модель работы функции. Для простоты будем рассматривать функцию одной переменной.

```
int f(int x) //шаг 1
{
++x; // шаг 2
return x; //шаг 3
}
…
y = f(y); //вызов функции
```

При вызове функции происходят следующие действия:

шаг 1*: int x = y*;
шаг 2: *++x*;
шаг 3: *NoName = x*;
шаг 4: y = *NoName*;

Если мы будем передавать не саму переменную, а её адрес, то сможем в функции работать с переменной. Например:

```
int* f(int* x) //шаг 1
{
++*x; // шаг 2
return x; //шаг 3
}
…
int* py = f(&y); //вызов функции
```

В этом примере при вызове функции происходят следующие действия:

шаг 1: *int* x = &y*; // В x мы положили адрес ячейка переменной y;

шаг 2: ++*x; // С помощью * мы обратились к x и увеличили его на единицу;

шаг 3: *int *py = x*; //В py мы положили адрес *x*.

Ещё одной важной возможностью влияние на значение переменной внутри функции является передача её по ссылке или псевдониму. Рассмотрим этот метод:

```
void f(int &x) //шаг 1
{
++x; // шаг 2
}
…
f(y); //вызов функции
```

При вызове функции происходят следующие действия:

шаг 1: *int &x = y*; //*x* объявляется как псевдоним *y*;

шаг 2: ++*x*; // *x* (*y*) увеличивается на единицу.

Передача параметров по ссылке также используется для работы в функциях с пользовательскими типами данных: структурами и классами. Дело в том, что для этих типов данных операция присваивания может быть не определена или крайне трудоёмка, поэтому используется передача переменной по ссылке, а для того, чтобы переменная не была изменена, псевдоним объявляется константным. Объявление такой функции имеет вид:

type f(*const type &x*),

где *type* – некоторый пользовательский тип данных.

Адресное выражение, стоящее перед скобками определяет адрес вызываемой функции. Это значит, что функция может быть вызвана через указатель на функцию.

Пример:

int (**fun*)(*int x, int *y*);

Здесь объявлена переменная *fun* как указатель на функцию с двумя параметрами: типа *int* и указателем на *int*. Сама функция должна возвращать значение типа *int*. Круглые скобки, содержащие имя указателя *fun* и признак указателя *, обязательны, иначе запись *int *fun* (*int x,int *y*); будет интерпретироваться как объявление функции *fun* возвращающей указатель на *int*. Вызов функции возможен только после инициализации значения указателя *fun* и имеет вид: (**fun*)(*i,&j*). В этом выражении для получения адреса функции, на которую ссылается указатель *fun* используется операция разыменования. Указатель на функцию может быть передан в качестве параметра функции.

Перегрузка функций – возможность использования одноимённых функций, с различным функционалом в языках программирования.

Две функции с одинаковым именем будут работать корректно, если они отличаются количеством формальных аргументов или их типом.

Пример:

int min(*int a, int b*) {*return a>b ? b : a*;}

int min(*int a, int b, int c*) {return (*a>b*)&&(*c>b*) ? b : min(*a, c*);} // не рекурсия! Обращение к другой //функции с тем же именем!

double min(*double a, double b*) {*return a>b ? b : a*;}

Функции, отличающиеся типом возвращаемого значения перегруженными не являются.

Пример:

int min(*int a, int b*) {*return a>b ? b : a*;}

long long int min(*int a, int b*) {*return a>b ? b : a*;}

В результате компилирования данного примера вернётся ошибка, говорящая об объявлении двух функций с одним и тем же именем.

Шаблоны (англ. *template*) — средство языка C++, предназначенное для кодирования обобщённых алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию).

Шаблоны позволяют создавать параметризованные классы и функции. Параметром может быть любой тип или значение одного из допустимых типов данных.

Например, нам необходимо получить в программе возможность взятия минимума от двух чисел независимо от типа данных. Мы можем воспользоваться перегрузкой и решить задачу следующим образом:

int min(*int a, int b*) {*return a>b ? b : a*;}

long long int min(*long long int* a, *long long int b*) {*return a>b ? b : a*;}

long double min(*long double* a, *long double* b) {*return a>b ? b : a*;}

float min(*float a, float b*) {*return a>b ? b : a*;}

Однако легко заметить, что тело всех функций одинаково. Переписывать одно и тоже кажется бессмысленно. Шаблон позволяет избежать этого. Для организации шаблона необходимо воспользоваться ключевым словом template. Всю рассмотренную выше перегрузку функций можно заменить на следующий шаблон:

template <typename Type >

Type min(*Type a, Type b*) {*return a>b ? b : a*;}

Теперь функцию min можно вызывать для переменных любого типа, для которого определена операция «*>*». Слово Type является пользовательским словом, вместо которого в процессе компилирования Си поставит нужный тип данных. Требование к имени типа такие же как и к имени переменных. Вместо слова *typename* можно использовать слово *class* – они равнозначны. Вызов шаблонной функции может происходить как и вызов обычной:

c = min(*a, b*)

или с непосредственным указанием типа:

c = min<int>(a, b).

При вызове функции без указания типа – тип берётся из первой переменной, на месте шаблонной. Это может породить следующую ошибку:

c = min(12,11.43)

при таком вызове компилятор либо детектирует ошибку, либо округлит 11.43 до *int*. Правильный вызов в таком случае имеет вид:

c = min<double>(12,11.43).

Если неизвестных типов несколько, то в скобках после *template* они перечисляются через запятую с ключевым словом *typename* вначале. При обработке шаблона компилятор, встретившись в программе с вызовом шаблонной функции с некоторым типом впервые – перегружает её, заменяя слово *Type* на имя соответствующего типа данных.

Задание на лабораторную работу

Объявить функции в соответствии с вариантом. Определить их. Написать демонстрационную программу. Демонстрационная программа – это программа, использующая все разработанные участки кода в процессе своего функционирования и позволяющая продемонстрировать все предусмотренные возможности разработанного кода.

Варианты лабораторных работ:

1. Написать функцию, которая добавляет элемент в конец массива состоящего из элементов произвольного типа. В качестве аргументов в функцию передаётся указатель на массив, количество элементов массива, значение элемента для вставки.
2. Написать рекурсивную функцию, вычисляющую значение функции Аккермана для натуральных *m, n.*
3. Написать функцию, которая добавляет элемент в начало массива состоящего из элементов произвольного типа. В качестве аргументов в функцию передаётся указатель на массив, количество элементов массива, значение элемента для вставки.
4. Написать рекурсивную функцию, вычисляющую значение функции вида:

$$F(x)=\begin{cases} 1, x=0, \\ x\cdot F(-x+1), x>0, \\ -x\cdot F(-x-1), x<0; \end{cases}$$

для целого значения x.

5. Написать функцию, которая добавляет элемент в массив, состоящий из элементов произвольного типа. В качестве аргументов в функцию передаётся: указатель на первый элемент массива, количество элементов массива, значение элемента для вставки и место вставки. Место вставки может передаваться двумя способами: с помощью индекса элемента, после которого необходимо провести вставку, или итератора, указывающего на него.
6. Написать функцию, меняющую каждый элемент массива на значение в нём другой функции, переданной первой в качестве аргумента. При написании демонстрационной программы проверить работу данной функции цикле с помощью массивов функций. Должна существовать возможность передать массив двумя способами: указатель на первый элемент массива и количество элементов массива или указатель на первый элемент массива (*beg*) и на «следующий за последним» (*end*), то есть обработка массива должна проводиться в полуинтервале [*beg, end*).
7. Написать функцию с переменным число аргументов, которая создаёт массив произвольного типа данных, заданной длины и забивает его переданными в качестве аргументов элементами.
8. Написать функцию, вычисляющую определитель квадратной матрицы элементов произвольного типа.
9. Написать функцию, добавляющую в конец массива некоторое произвольное число элементов. Элементы могут быть переданы в функцию в виде другого массива или от пользователя в качестве аргументов функции с переменным числом аргументов.
10. Написать функцию, находящую первый элемент массива, для которого переданная ей в качестве аргумента функция возвращает значение *true*.
11. Написать функцию с переменным числом аргументов, находящую первый из переданых ей пользователем в качестве аргументов элементов, для которого переданная ей в качестве аргумента функция возвращает значение *true* и аналогичную функцию, но ищущую первое значение в переданном массиве.
12. Написать функцию, удаляющую элементы массива произвольного типа данных, для которых переданная ей в качестве аргумента функция возвращает значение *true*.
13. Написать функцию, удаляющую все нулевые элементы массива произвольного типа данных.
14. Написать функцию *resize*(), меняющую размер массива произвольного типа данных. Функция принимает в качестве аргумента указатель на массив, количество элементов массива, новое число элементов.
15. Написать функцию удаляющую элемент массива произвольного типа данных. Элемент может быть указан как с помощью индекса, так и с помощью итератора. Функция также принимает указатель на массив и количество элементов массива.
16. Написать функцию, которая осуществляет бинарный поиск на упорядоченном полуинтервале. В качестве аргументов передавать итераторы на первый и следующий за последним элемент, а также искомое значение.
17. Написать функцию, которая проходя полуинтервал, заданный итераторами заменяет каждое значение, удовлетворяющее условию на переданное в качестве аргумента. Условие также передаётся в качестве аргумента с помощью указателя на функцию.

18. Написать функцию, проверяющую отсортирован ли заданный полуинтервал массива. Полуинтервал задаётся итераторами. Функция возвращает *true*, если все элементы интервала стоят упорядоченными от меньшего к большему и *false* в противном случае.

19. Написать функцию, вычисляющую выражения $\underbrace{f(f(f(...f(x))))}_{n\ вызовов\ f}$. Функция принимает в качестве аргументов *n*, *f* и *x*.

20. Написать функцию, которая в качестве аргументов принимает массив функций (указатель и размер) $f_1,\ f_2,\ ...,\ f_n$ или набор этих функций в качестве аргументов, а также некоторую переменную *x* и возвращает значение выражения: $f_1(f_2(f_3(...f_n(x))))$.

21. Напишите функцию, вычисляющую выражение:

$$1-\cfrac{2}{3+\cfrac{4}{5-\cfrac{6}{7+\cfrac{8}{\cfrac{...}{(2\cdot n-1)+(-1)^n\cfrac{2\cdot n}{2\cdot n+1}}}}}};$$

без использования цикла, где *n* – некоторое натуральное число, вводимое в качестве аргумента функции.

22. Напишите функцию, вычисляющую выражение: $1+\cfrac{x^2}{2+\cfrac{x^4}{3+\cfrac{x^6}{4+\cfrac{...}{n+\cfrac{x^{2n}}{n+1}}}}}$;

без использования цикла, где *n* – некоторое натуральное число, а *x* – вещественное. Оба числа вводятся в качестве аргумента функции.

23. Напишите функцию, вычисляющую выражение: $sh(ch(2\cdot sh(3\cdot ch(...\cdot 2\cdot n\cdot shx...))))$ без использования цикла, где *n* – некоторое натуральное число, а *x* – вещественное. Оба числа вводятся в качестве аргумента функции.

24. Напишите функцию, вычисляющую выражение: $tg(1\cdot\frac{\pi}{2}+tg(2\cdot\frac{\pi}{2}+tg(3\cdot\frac{\pi}{2}+...+tg(n\cdot\frac{\pi}{2}+x)...)))$;

без использования цикла, где *n* – некоторое натуральное число, а *x* – вещественное. Оба числа вводятся в качестве аргумента функции.

25. Напишите функцию, вычисляющую выражение: $\sin(\cos(\ 2\cdot\sin(\ 3\cdot\cos(...\cdot 2\cdot n\cdot\sin\ x...))))$ без использования цикла, где *n* – некоторое натуральное число, а *x* – вещественное. Оба числа вводятся в качестве аргумента функции.

26. Напишите функцию, вычисляющую выражение: $\lg(1+\lg(\ 2+\lg(\ 3+...+\lg(\ n+x)...)))$; без использования цикла, где *n* – некоторое натуральное число, а *x* – вещественное. Оба числа вводятся в качестве аргумента функции.

27. Напишите функцию, вычисляющую выражение: $\sin(\ x^n+\sin(\ x^{n-1}+\sin(\ x^{n-2}+...+\sin(\ x)...)))$; без использования цикла, где *n* – некоторое натуральное число, а *x* – вещественное. Оба числа вводятся в качестве аргумента функции.

***Примечание* 1. Если в задании говорится, что функция в качестве аргументов может принимать либо одно, либо другое, то от Вас требуется решить это путём перегрузки функции, то есть каждое «либо» увеличивает число требуемых к написанию функций.**

***Примечание* 2. В некоторых заданиях может понадобиться написание вспомогательных функций.**

***Примечание 3*. Если ваше задание относительно элементов «произвольного типа» это значит, что функция должна работать относительно переменных любого типа.**

Распределение Вариантов

№	ФИО	Вариант
1	Алпацкая Юлия Максимовна	13
2	Белова Юлия Андреевна	14
3	Бредихина Дарья Алексеевна	15
4	Денисов Лев Сергеевич	21
5	Дронов Артем Романович	18
6	Казаков Александр Александрович	24
7	Коблов Никита Олегович	23
8	Ковалёв Арсений Дмитриевич	22
9	Ломакин Сергей Александрович	17
10	Мешков Виталий Владимирович	12
11	Назаров Григорий Алексеевич	11
12	Осипова Ксения Александровна	9
13	Праздников Ярослав Альбертович	10
14	Рагулина Виктория Викторовна	25
15	Рачицкий Вадим Витольдович	26
16	Самойлов Максим Николаевич	5
17	Сафронов Иван Юрьевич	8
18	Соболев Станислав Юрьевич	6
19	Сысоева Анна Германовна	3
20	Токарев Антон Сергеевич	2
21	Трофимчук Роман Алексеевич	1
22	Чижов Кирилл Дмитриевич	7
23	Шибков Даниил Дмитриевич	4
24	Щербаков Артём Анатольевич	19
25	Якушев Олег Александрович	17