Практическое задание 8
Использование функций в С++

Цель. Изучение принципов модульной технологии разработки программ и использование их в своем программировании. Изучение синтаксических правил описания функций и обращения к ним. Изучение механизмов обращения к функции, передачи данных в функцию и механизма возвращения данных. Перегрузка функций.

ПЛАН

1. Краткое теоретическое введение 1
2. Передача параметров по ссылке 2
3. Перегруженные функции 3
4. Область действия и время жизни имен. Статические объекты 3
Бонус 3

1. Краткое теоретическое введение

Общепринято описания функций выносить в отдельные файлы. Описание функции порождает программную единицу, самостоятельную и независимую.

К любой функции можно обратиться из любой другой функции оператором обращения (вызова). При этом происходит такая цепь событий.

1. Управление передается в функцию по оператору обращения.
2. Выделяется память для параметров функции, вычисляются их значения и копируются в локальную память.
3. Создаются локальные переменные функции, которые живут только в теле функции.
4 Выполняется алгоритм функции, использующий внешние и локальные данные.
5. По `return` управление передается в точку вызова, в вызывающую программу передается возвращаемое значение.
6. Локальные переменные умирают, память высвобождается.

Все это процессы хорошо видны в отладчике при пошаговом исполнении программы.

Параметры по ссылке

Функция С++ возвращает одно значение. Если необходимо, чтобы функция вычисляла и возвращала более одного значения, используется **передача параметров по ссылке**. Знак &, это признак адресной операции. Запись &A, где A – имя переменной, означает, что получен адрес объекта A в оперативной памяти.

Описание параметра по ссылке имеет синтаксис:

```
Тип & Имя_параметра
```

Функция и вызывающая программа работают с адресом объекта в памяти (с одной и той же областью данных, выделенной объекту в вызывающей программе), следовательно:

1) параметр является единым объектом для функции и для вызывающей программы;
2) функция может изменить значения переданных ей параметров;
3) фактический параметр, передаваемый по ссылке, может быть только адресуемым данным, это переменная.

Область действия и время жизни имен

Программные объекты в коде, это переменные, именованные константы, функции. Все они представлены своим именем.

Область действия – это область программного кода, в которой объект известен (то есть действует его объявление).

Время жизни – понятие, связанное с областью действия, это период времени в процессе выполнения программы, когда объект фактически занимает память.

Принцип локализации имен означает, что каждый объект кода существует только

внутри того блока, в котором объявлен, например:

```
for (int i=0; i<=5; i++)
{
… // Имя i известно только в теле цикла.
}
// При выходе из цикла переменная i не существует более.
```

Память под объект распределяется при входе в блок, высвобождается при завершении цикла. Это значит, что область действия – только тело цикла, а время жизни – время выполнения циклического алгоритма.

Тот же принцип распространяется и на параметры функции. Можно считать, что параметры по значению являются для функции локальными, а параметры по ссылке – глобальными.

Статические переменные

Статические переменные порождаются в теле функции, но имеют глобальное время жизни, предваряются ключевым словом `static` в описании:

```
static Тип Имя;
```

Как глобальное имя смысла не имеет, но будучи объявлено в теле функции**:**

- получает память один раз при старте программы;
- обнуляется;
- при повторных входах в функцию значение сохраняется.

2. Передача параметров по ссылке

Заголовочный файле `Header.h` скопируйте в папку проекта и присоедините к проекту, откройте и ознакомьтесь с содержимым. Все описания функций выполняются здесь, а управления вызовами – в `Source` файле проекта.

Упражнение 1. Параметры по ссылке

Функция, которая возвращает более одного значения, должна это делать через параметры. Например, нужна функция, которая меняет значения двух переменных. Для этого описана функция **`Swap1(x, y)`**. Прочтите ее текст. Напишите обращение:

```
int a=5, b=10;
cout <<"a=" << a << " b=" << b<< endl;
Swap1 (a, b);
cout <<"a=" << a << " b=" << b<< endl;
```

Выполните в отладчике. Посмотрите, что произошло, объясните результат.

А теперь прочтите текст функции **`Swap2(x, y)`**, обратитесь к ней таким же образом, выполните в отладчике, объясните результат.

Передайте функции **`Swap1`** значения констант 3 и 7. Попробуйте передать константы функции **`Swap2`** . Объясните различие.

Устно ответьте на вопросы.

1. Синтаксическое отличие передачи параметров по адресу от передачи по значению.
2. Отличие механизма передачи параметров по адресу от передачи по значению.

Упражнение 2. Задание для самостоятельной разработки.

Опишите функцию **`Range`** с тремя параметрами, которая меняет значения параметров так, чтобы они были упорядочены по возрастанию.

Обратитесь к ней.

3. Перегруженные функции

Примером функции типа **void** не возвращающей значения, является функция **print()**, которая выводит символы на экран. Напишите вызывающую программу, которая обращается к функции **print()**.

Упражнение 3. Перегрузка функций.

В этом же заголовочном файле перегрузите функцию **print(int n)** с параметром, которая выводит на экран указанное число символов '*'. Обратитесь к ней 2-3 раза с разными значениями.

В этом же заголовочном файле перегрузите функцию **print(int n, char c)** с двумя параметрами, которая выводит на экран указанное число указанных символов. Обратитесь к ней 2-3 раза с разными значениями фактических параметров. Добавьте в последнюю функцию контроль над входными данными, а именно, если фактическое значение n>80, то пусть выводится ровно 80 символов.

4. Область действия и время жизни имен. Статические объекты

Упражнение 4. Исследование принципа работы статического объекта

Опишите в заголовочном файле две функции.

```
void Auto_f(void)
{    int K = 1;
     cout << "\tK= " << K << endl;
     K++;
}
void Stat_f(void)
{    static int K = 1;
     cout << "\tK= " << K << endl;
     K++;
}
```

Обратитесь к ним так:

```
for (int i=1; i<=5; i++)
     Auto_f();
for (i=1; i<=5; i++)
     Stat_f();
```

Выполните в отладчике по шагам, наблюдая значение переменной **K**.

Ответьте на вопрос.

1. Как влияет класс **static** на поведение объекта?

Бонус

1. Опишите функцию, находящую сумму двух простых дробей, заданных значениями числителей и знаменателей.

Обратитесь к функции в диалоге.

2. Опишите функцию, которая может вычислять таблицу любой зависимости Y(x). Для этого нужно научиться передавать функцию в функцию.