Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

ПЕРЕГРУЗКА ФУНКЦИЙ. ШАБЛОНЫ ФУНКЦИЙ.

Отчет по лабораторной работе по дисциплине «Объектно-ориентированное программирование»

Студент группы 549
 Баулин С.К.
«__» _____ 2020 г.
Кандидат физико-математических наук, доцент кафедры ЭМИС
_____ _____ Шельмина Е. А.
оценка «__» _____ 2020 г.

Томск 2020

Лабораторная работа №3

Перегрузка функций. Шаблоны функций.

Цель работы

Ознакомиться с возможностью перегрузки функций и научиться применять полученные знания на практике. Научиться использовать шаблоны функций и функции с переменным количеством параметров.

Краткий теоретический материал

Каждая программа на C++ – это совокупность функций, каждая из которых должна быть определена или описана до её использования в конкретном модуле программы. Рассмотрим более сложные примеры использования функций.

Перегрузка функций

Перегрузкой функций называется использование нескольких функций с одним и тем же именем, но с различными списками параметров. Перегруженные функции должны отличаться друг от друга либо типом хотя бы одного параметра, либо количеством параметров, либо и тем и другим одновременно. Перегрузка является видом полиморфизма и применяется в тех случаях, когда одно и то же по смыслу действие реализуется по-разному для различных типов или структур данных. Компилятор сам определяет, какой именно вариант функции вызвать, руководствуясь списком аргументов.

Если же алгоритм не зависит от типа данных, лучше реализовать его не в виде группы перегруженных функций для различных типов, а в виде шаблона функции. В этом случае компилятор сам сгенерирует текст функции для конкретных типов данных, с которыми выполняется вызов, и программисту не придется поддерживать несколько практически одинаковых функций.

Небольшие перегруженные функции удобно применять при отладке программ.

При написании перегруженных функций основное внимание следует обращать на то, чтобы в процессе поиска нужного варианта функции по ее вызову не возникало неоднозначности.

Неоднозначность может возникнуть по нескольким причинам. Во-первых, из-за преобразований типов, которые компилятор выполняет по умолчанию. Их смысл сводится к тому, что более короткие типы преобразуются в более длинные. Если соответствие между формальными параметрами и аргументами функции на одном и том же этапе может быть получено более чем одним способом, вызов считается неоднозначным и выдается сообщение об ошибке.

Неоднозначность может также возникнуть из-за параметров по умолчанию и ссылок.

Преимущества перегрузки функции:

- перегрузка функций улучшает удобочитаемость программ;
- перегрузка функций C++ позволяет программам определять несколько функций с одним и тем же именем;
- перегруженные функции возвращают значения одинакового типа, но могут отличаться количеством и типом параметров;
- перегрузка функций упрощает задачу программистов, требуя, чтобы они помнили только одно имя функции, но тогда они должны знать, какая комбинация параметров соответствует какой функции.

Рекурсивные функции

Любая функция в программе на C++ может вызываться рекурсивно. При этом в стеке выделяется новый участок памяти для размещения копий параметров, а также автоматических и регистровых переменных, поэтому предыдущее состояние выполняемой функции сохраняется, и к нему впоследствии можно вернуться (так и произойдет, если только ваша программа где-нибудь не зависнет).

Различают прямую и косвенную рекурсию. Функция называется косвенно-рекурсивной в том случае, если она содержит обращение к другой функции, содержащей прямой или косвенный вызов определяемой первой функции. Если в теле функции используется явный вызов этой же функции, то это прямая рекурсия.

Однако у рекурсии есть и недостатки: во-первых, такую программу труднее отлаживать, поскольку требуется контролировать глубину рекурсивного обращения, во-вторых, при большой глубине стек может переполниться, а в-третьих, использование рекурсии повышает

Шаблоны функций

Цель введения шаблонов функций – автоматизация создания функций, которые могут обрабатывать разнотипные данные. В отличие от механизма перегрузки, когда для каждой сигнатуры определяется своя функция, шаблон семейства функций определяется один раз. Шаблон располагается перед main.

```
template <class type>
type имя_функции (список_формальных_параметров)
{
тело функции
}
```

Здесь type – любой корректный идентификатор, который автоматически заменяется компилятором на любой стандартный тип.

При использовании шаблонов уже нет необходимости готовить заранее все варианты функций с перегруженным именем. Компилятор автоматически, анализируя вызовы функций в тексте программы, формирует необходимые определения именно для таких типов параметров, которые использованы в обращениях. Дальнейшая обработка выполняется так же, как и для перегруженных функций.

Основные свойства параметров шаблона:

1. Имена параметров шаблона должны быть уникальными всем определении шаблона.

2. Список параметров шаблона функции не может быть пустым, так как при этом теряется возможность параметризации и шаблон функции становиться обычным определением конкретной функции.

3. В списке параметров шаблона функции может быть несколько параметров. Каждый из них должен начинаться со служебного слова class.

Допустимый заголовок шаблона:

template <class type1, class type2>

Соответственно, неверен заголовок:

template <class type1, type2, type3>

4. Недопустимо использовать в заголовке шаблона параметры с одинаковыми именами, т.е. ошибочен такой заголовок:

template <class type1, class type1, class type1>

Функции с переменным количеством параметров

В C++ допустимы функции, у которых количество параметров при компиляции определения функции не определено. Кроме того, могут быть неизвестными и типы параметров. Количество и типы параметров становятся известными только в момент вызова функции, когда явно задан список фактических параметров. При определении и описании таких функций спецификация формальных параметров заканчивается многоточием:

тип имя (список_явных_параметров, ...);

Каждая функция с переменным списком параметров должна иметь один из двух механизмов определения их количества и типов.

Первый подход предполагает добавление в конец списка реально использованных, необязательных, фактических параметров специального параметра-индикатора с уникальным значением которое будет сигнализировать об окончании списка. В теле функции параметры

перебираются, и их значение сравнивается с заранее известным концевым признаком.

Второй подход предусматривает передачу в функцию значения реального количества используемых фактических параметров, которые передаются с помощью одного из обязательных параметров.

В обоих случаях переход от одного фактического параметра к другому выполняется с помощью указателей (с использованием адресной арифметики).

Признаком окончания списка фактических параметров служит параметр с нулевым значением.

Чтобы функция с переменным количеством параметров воспринимала параметры различных типов, то для однотипных параметров необходимо передавать информацию функции с помощью обязательного дополнительного параметра.

Задания

Задание 1. Определить функцию, возвращающую количество дней до конца месяца. Выполнить перегрузку функции для следующих типов параметров: Структура «дата» (год, месяц, день). Три целочисленных параметра: год, месяц, день. Два целочисленных параметра: месяц, день (считать передаваемые числа датой текущего года).

Задание 2. На основе задания 1 построить шаблон функции

Ход работы

Задание 1. Код и результат выполненного задания показаны на рисунках 1 – 3

Задание 2. Код и результат выполненного задания показаны на рисунке 4

Вывод

Ознакомлены с возможностью перегрузки функций и научиться применять полученные знания на практике. Научены использовать шаблоны функций и функции с переменным количеством параметров.

```
int days_left(date data)
{
    int n = 0;
    switch (data.month)
    {
    case 1:
        n = 31;
        break;
    case 2:
        if (data.year % 4 == 0 && data.year % 100 != 0 || data.year % 400 == 0) // високосный год или нет
            n = 29;
        else
            n = 28;
        break;
    case 3:
        n = 31;
        break;
    case 4:
        n = 30;
        break;
    case 5:
        n = 31;
        break;
    case 6:
        n = 30;
        break;
    case 7:
        n = 31;
        break;
    case 8:
        n = 31;
        break;
    case 9:
        n = 30;
        break;
    case 10:
        n = 31;
        break;
    case 11:
        n = 30;
        break;
    case 12:
        n = 31;
        break;
    default:
        break;
    }

    return (n - data.day);
}
```

Рисунок 1 – Скриншот перегрузки функции для структуры date

```cpp
int days_left(int year, int month, int day)
{
    int n = 0;
    switch (month)···

    return (n - day);
}

int days_left(int month, int day)
{
    time_t t;
    struct tm *timeN;
    time(&t);
    timeN = localtime(&t);
    int yearNow = timeN->tm_year + 1900;

    int n = 0;
    switch (month)···
    return (n - day);
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    setlocale(LC_ALL, "Russian");
    date data;
    int year, month, day;

    cout << "Перегрузка функции для структуры: \nВведите год: ";
    cin >> data.year;
    cout << "Месяц: ";
    cin >> data.month;
    cout << "День: ";
    cin >> data.day;
    cout << "До конца месяца осталось: " << days_left(data) << " дней\n"
        << endl;

    cout << "Перегрузка функции с 3 параметрами: \nВведите год: ";
    cin >> year;
    cout << "Месяц: ";
    cin >> month;
    cout << "День: ";
    cin >> day;
    cout << "До конца месяца осталось: " << days_left(year, month, day) << " дней\n"
        << endl;

    cout << "Перегрузка функции с 2 параметрами: \nВведите месяц: ";
    cin >> month;
    cout << "День: ";
    cin >> day;
    cout << "До конца месяца осталось: " << days_left(month, day) << " дней" << endl;

    return 0;
```

Рисунок 2 – Скриншот кода задания 1

```
Перегрузка функции для структуры:
Введите год: 1992
Месяц: 10
День: 25
До конца месяца осталось: 6 дней

Перегрузка функции с 3 параметрами:
Введите год: 2025
Месяц: 3
День: 14
До конца месяца осталось: 17 дней

Перегрузка функции с 2 параметрами:
Введите месяц: 2
День: 27
До конца месяца осталось: 2 дней
```

Рисунок 3 – Скриншот результата задания 1

```
18  template <class T>
19  T days_left(T year, T month, T day)
20  {
21      int n = 0;
22      // определение количества дней
23  >   switch (month)···
67      return (n - day);
68  }
69
70  int main()
71  {
72      SetConsoleCP(1251);
73      SetConsoleOutputCP(1251);
74      setlocale(LC_ALL, "Russian");      You, 23 days ago • laba3 edit
75      date data;
76      int year, month, day;
77
78      cout << "Перегрузка функции для структуры: \nВведите год: ";
79      cin >> data.year;
80      cout << "Месяц: ";
81      cin >> data.month;
82      cout << "День: ";
83      cin >> data.day;
84      cout << "До конца месяца осталось: " << days_left(data.year, data.month, data.day) << " дней\n"
85           << endl;
86
87      cout << "Перегрузка функции с 3 параметрами: \nВведите год: ";
88      cin >> year;
89      cout << "Месяц: ";
90      cin >> month;
91      cout << "День: ";
92      cin >> day;
93      cout << "До конца месяца осталось: " << days_left(year, month, day) << " дней\n"
94           << endl;
95

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  КОНСОЛЬ ОТЛАДКИ

PS C:\Users\seron\Desktop\study\3sem\OOP\laba3> .\laba3_2.exe
Перегрузка функции для структуры:
Введите год: 1992
Месяц: 10
День: 25
До конца месяца осталось: 6 дней

Перегрузка функции с 3 параметрами:
Введите год: 2001
Месяц: 1
День: 1
До конца месяца осталось: 30 дней
```

Рисунок 4 – Скриншот выполненного задания 2