УДК 004

РАЗНИЦА МЕЖДУ ОБЪЯВЛЕНИЕМ И ОПРЕДЕЛЕНИЕМ В *C* И *C++*

Гагарин Вячеслав Юрьевич
студент
Мордовский государственный университет им. Н.П. Огарева, Саранск

author@apriori-journal.ru

Аннотация. В данной статье рассматривается различие между «определением» и «объявлением» класса, переменной или функции в языках C/C++; рассказывается о том, как избежать возникновения нестандартных ошибок из-за неправильного «определения» или «объявления» класса, переменной или функции.

Ключевые слова: объявление, определение, C, C++, ошибка, extern, класс, функция, переменная.

DIFFERENCE BETWEEN THE ANNOUNCEMENT AND DEFINITION IN *C* AND *C ++*

Gagarin Vyacheslav Yuryevich
student
Ogarev Mordovia State University, Saransk

Abstract. In this article is considered distinction between «definition» and «announcement» of a class, variable or function; describes, how to avoid non-standard errors due to incorrect «definitions» or «announcement» of a class, variable or function.

Key words: announcement, definition, C, C++, error, extern, class, function, variable.

Если говорить об истории развития этих языков, то первоначально появился язык C, а уже впоследствии как развитие языка C появился и C++. Язык C оказал большое влияние на индустрию разработки программного обеспечения. С одной стороны, синтаксис многих его инструкций лежит в основе таких языков, как C++, C#, Java, PHP. С другой, он используется в качестве промежуточного в некоторых системах программирования, когда программа сначала транслируется в программу на языке C и только потом компилируется компилятором языка C для получения окончательного исполняемого модуля [1]. Для аналогичных целей используется и язык C++, но его важным отличием от языка C является поддержка технологии ООП.

В C и C++ существует одно маленькое, но важное различие между «объявлением» и «определением» класса, переменной или функции. Необходимо понимать разницу между значением этих слов, иначе вы часто будете натыкаться на странные ошибки, такие как:

undefined reference to vtable for foo (в C++)

или

undefined symbol foo

или

undefined reference to foo

При объявлении класса, функции или переменной, вы сообщаете компилятору, что есть что-то определенного типа и с определенным именем. Но компилятор сможет обработать большую часть использований данного имени без необходимости полного определения этого имени. То есть можно объявить то или иное значение, не определив его. Это позволяет писать код, понятный компилятору, опуская дополнительные детали. Удобно это в том случае, если при работе с несколькими исходными файлами, вы должны использовать функцию в нескольких

файлах. То есть можно объявить функцию, но писать тело функции в каждом из файлов не нужно.

Иными словами, «объявление» выглядит так:

```
double function();
```

Оно не включает тело функции, но сообщает компилятору, что она (функция) будет определена где-то позже, поэтому компилятор может использовать её.

А вот «определение» означает описание всей необходимой информации для создания того, что мы определяем, целиком. Описание всех полей, методов и свойств класса есть «определение» класса. Описание тела функции есть «определение» функции. Конечно, вы можете как объявлять классы, переменные или функции, так и определять их одновременно. Но вам это не нужно, потому что, как только что-то определено, оно считается и объявленным.

Для компилятора часто достаточно просто «объявления».

Например:

```
double function();
  int main()
    {
      double a = function();
    }
double function()
  {
    return 2.5;
  }
```

Количество аргументов, которые принимает функция, а также возвращаемое значение, компилятор знает, поэтому он может скомпилировать вызов функции, даже если она еще не определена, или определение функции находится в другом файле.

Объявим класс, не определяя его:

```
class MyFirstClass;
```

Попробуем узнать, что находится в MyFirstClass. Напишем следующий код:

```
class MyFirstClass;
 MyFirstClass aza_obj;
class MyFirstClass
 {
   double MyField;
 };
```

Данный фрагмент кода вызовет ошибку, потому что компилятор не знает размер переменной aza_obj, а из «объявления» MyFirstClass он его узнать не может; ему нужно «определение», которое появляется ниже.

В большинстве случаев, когда вы объявляете переменную, вы также определяете ее. Поэтому «определение» переменной означает, что вы говорите компилятору, где выделить место для хранения этой переменной.

Напишем такой код:

```
double x;
int main()
 {
  x = 3.5;
 }
```

Строка double x и определяет, и объявляет переменную. На более простом языке это означает: «создать переменную с именем x, типа double». При этом место, где хранится переменная, определяется тем, что это глобальная переменная, определенная в объектный файл, связанный с этим исходным файлом.

Создадим второй исходный файл, имеющий такой исходный код:

```
extern double x;
int function()
 {
   x = 3.5;
 }
```

Использование extern позволяет объявить переменную, не определяя ее, extern сообщает, что переменная находится где-то в другом месте. Спецификатор extern играет большую роль в программах, состоящих из многих файлов. В языках C/C++ программа может быть записана в нескольких файлах, которые компилируются раздельно, а затем компонуются в одно целое. В этом случае необходимо как-то сообщить всем файлам о глобальных переменных программы. Самый лучший (и наиболее переносимый) способ сделать это – определить (описать) все глобальные переменные в одном файле и объявить их со спецификатором extern в остальных файлах [2].

Напишем следующий код:

```
extern double x;
int function()
 {
   x = 3.5;
 }
double x;
```

В данном примере «определение» переменной x находится внизу, а «объявление» в верхней части программы.

Если в заголовке файла поставить переменную и не использовать extern, то возникнет обратная задача неопределенного символа [3], то есть у вас будет символ с несколькими определениями с ошибкой типа:

redefinition of foo

Это произойдет, когда компоновщик будет связывать объектные файлы.

В общем случае «объявление» предоставляет основные свойства символа: название и тип. «Определение» предоставляет все детали этого символа – если это класс, то какие у него методы, поля и свойства; если это функция, то что она вычисляет; если это переменная, то где эта переменная хранится. Часто компилятору нужно «объявление», чтобы скомпилировать файл в объектный файл, так как компоновщик может найти «определение» из другого файла. Если исходный файл объявлен, но символ он не определяет, то во время компоновки будут возникать ошибки, сообщающие о неопределенных символах [4].

Если вы хотите использовать функцию в нескольких исходных файлах, вы должны объявить функцию в одном заголовочном файле (.h), а затем определить функцию в одном исходном файле (.c или .cpp). Весь код, который использует функция должен включать в себя только .h файл, и вам надо связать полученные объектные файлы с объектным файлом от компиляции исходного файла.

Если вы хотите использовать переменную в нескольких файлах, вы должны объявить переменную с помощью ключевого слова extern в одном заголовочном файле, а затем включить этот заголовочный файл во все исходные файлы, которым нужна эта переменная. Затем вы должны

определить эту переменную в одном исходном файле, который связан со всеми объектными файлами, которые используют эту переменную.

Если вы хотите использовать класс в нескольких файлах, вы должны поместить «определение» класса в заголовочный файл и определить методы класса в соответствующем исходном файле.

Список использованных источников

1. Александров Э.Э., Афонин В.В. Введение в программирование на языке С. Саранск: Изд-во Мордов. ун-та, 2009. 5 с.
2. Александров Э.Э., Афонин В.В. Программирование на языке С в Microsoft Visual Studio 2010 [Электронный ресурс]. Режим доступа: http://www.intuit.ru/department/pl/prcmsvs2010 (дата обращения: 25.11.2015).
3. Таненбаум Э. Современные операционные системы. СПб., 2010. 972 с.
4. Прата С. Язык программирования С. Лекции и упражнения. М.: Вильямс, 2006. 256 с.