Сборка и объектные файлы

File Makefile:

```
CXX = g++
CXXFLAGS = -Wall -g

#
OBJECTS = main.o module1.o module2.o module3.o
SOURCES = main.cpp module1.cpp module2.cpp module3.cpp
HEADERS = module1.h module2.h module3.h

#
my_program: $(OBJECTS)
	$(CXX) -o my_program $(OBJECTS)

#
%.o: %.cpp $(HEADERS)
	$(CXX) $(CXXFLAGS) -c $< -o $@

#
clean:
	rm -f $(OBJECTS) my_program
```

Вывод программы

3. **Объяснение строк (A) и (D) в `main.cpp`:**

   - **(A)**

     ```
     std::cout << getMyName() << "\n"; // (A)
     ```

     Так как перед этой строкой было заключено using namespace Module1;, выполнится функция Module1::getMyName(), которая вернет "John".

   - (D)

     ```
     using Module2::getMyName;
     std::cout << getMyName() << "\n"; // (D)
     ```

     Здесь объявлено using Module2::getMyName;, поэтому вызывается функция Module2::getMyName(), которая вернет "James".

Ошибки компиляции

4. Раскомментируем строки (B) и (C):

```
using namespace Module2; // (B)
std::cout << getMyName() << "\n"; // COMPILATION ERROR (C)
```

Это приведет к ошибке компиляции. Ошибка происходит из-за неоднозначности вызова функции getMyName(), не ясно, использовать Module1::getMyName или Module2::getMyName.

Решение:

- Указывать, какую именно функцию вызывать:

```
std::cout << Module1::getMyName() << "\n";
std::cout << Module2::getMyName() << "\n";
```

Добавление новой функции

5. Добавим новую функцию getMyName() в еще одном пространстве имен:

   Создадим файлы `module3.h` и `module3.cpp`.

   File module3.h:

```
#include <string>

namespace Module3
{
  std::string getMyName();
}
```

   File module3.cpp:

```
#include "module3.h"

namespace Module3
{
  std::string getMyName()
  {
    return "Peter";
  }
}
```

   Изменим `main.cpp` для вызова новой функции:

```
#include "module1.h"
#include "module2.h"
#include "module3.h"
#include <iostream>

int main(int argc, char** argv)
{
```

```
  std::cout << "Hello world!" << "\n";

  std::cout << Module1::getMyName() << "\n";
  std::cout << Module2::getMyName() << "\n";

  using namespace Module1;
  std::cout << getMyName() << "\n"; // (A)
  std::cout << Module2::getMyName() << "\n";

  //using namespace Module2; // (B)
  //std::cout << getMyName() << "\n"; // COMPILATION ERROR (C)

  using Module2::getMyName;
  std::cout << getMyName() << "\n"; // (D)

  std::cout << Module3::getMyName() << "\n";
}
```

Избавление от `std::cout`

6. Чтобы избавиться от необходимости писать `std::cout`, можно использовать конструкцию using namespace std:

```
#include <iostream>

using namespace std;

int main()
{
  cout << "Hello world!" << endl;
  return 0;
}
```

Таким образом, можно использовать `cout` и `endl` без префикса `std::`.