Сборка и объектные файлы

1. **Соберtv программу**, чтобы увидеть, что на каждый `*.cpp` файл создаётся отдельный объектный файл. Для Visual Studio это будут файлы с расширением `*.obj`, и они будут созданы в папке Debug.

2. **Убедимся, что при изменении одного `*.cpp` файла и пересборке проекта обновляется только соответствующий ему объектный файл.** Для этого изменим код в `module1.cpp` или `module2.cpp`, пересоберем проект и проверим даты изменения объектных файлов.

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::`.