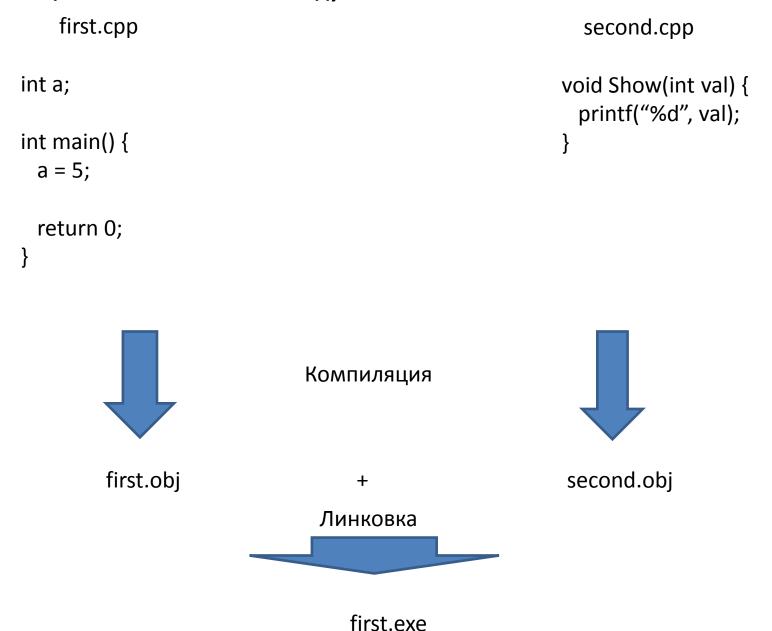
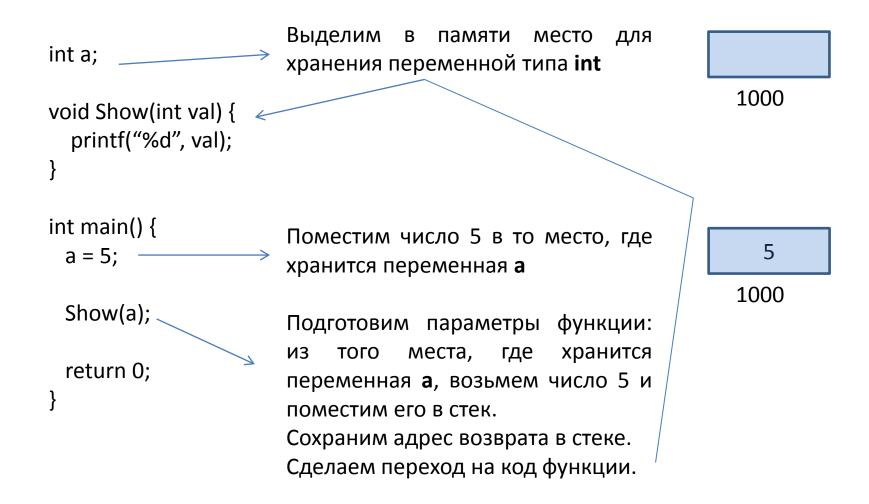
В языке С используется раздельная компиляция файлов: сначала каждый файл компилируется по отдельности и затем результаты (объектные модули) объединяются линковщиком в исполняемый модуль.



Если у нас есть только один файл, то работа с переменными происходит следующим образом:



В случае нескольких файлов они не видят переменные, объявленные в другом месте. Например, переменная а объявлена в первом файле и поэтому она недоступна во втором. При компиляции второго файла (а она проходит отдельно) компилятор не имеет информации о данной переменной и поэтому выдаст ошибку.

Компиляция

Спецификатор хранения extern подсказывает компилятору, что данная переменная объявлена где-то в другом месте и дает информации о ее типе.

Теперь компилятор может скомпилировать второй файл, подразумевая, что настоящий адрес переменной а будет затем подставлен линковщиком.

first.cpp second.cpp int a; _____ Выделим в памяти 4 байта для extern int a; хранения переменной типа **int** int main() { void Show(int val) { Поместим число 5 в то место, a = 5; a = 25; где хранится переменная а return 0; Сообщим компилятору, переменная **a** имеет тип **int** и определена где-то в другом месте Поместим число 25 в пока неизвестное место (подставим ???? вместо адреса и пометим это место в коде модуля)

Линковка: вместо ???? подставим реальный адрес переменной а, взятый из модуля first.cpp

```
first.cpp second.cpp

extern int a;

int main() {

return 0;

void Show(int val) {

a = 25;

}
```

Если переменная а не нигде не объявлена, то мы получим ошибку при линковке нашей программы.

Линковка: вместо ???? пытаемся подставить реальный адрес переменной а, но его нигде нету: ошибка линковки

Error LNK2019 unresolved external symbol a

```
first.cpp second.cpp

int a;

int a;

void Show(int val) {
 return 0;
 }

}
```

Линковка: в проекте дважды объявлена переменная а - ошибка линковки

Error LNK2005 "int a" (?a@@3HA) already defined

Спецификатор хранения static делает переменную локальной для данного модуля (файла) и позволяет использовать одноименные глобальные переменные в разных модулях. Следующая программа корректно скомпилируется и слинкуется.

В этом примере компилятор не имеет информации о функции Show при компиляции первого файла. Поэтому мы получим ошибку компиляции.

Сейчас мы даем компилятору информацию о функции, определяя ее прототип. Эта программа корректно скомпилируется.

```
first.cpp second.cpp

int a; void Show(int val) {
    printf("%d", val);

void Show(int val);
}

int main() {
    a = 5;
    Show(a);
    return 0;
}
```

Часто в больших проектах создают специальные заголовочные файлы, в которых перечисляют нужные различным модулям прототипы функций:

```
second.cpp
    first.cpp
                                                void Show(int val) {
                                                  printf("%d", val);
#include "funct.h"
int a;
int main() {
 a = 5;
 Show(a);
  return 0;
                           funct.h
                         void Show(int val);
```