OS DAY 2024

Мониторинг потока управления процессов в операционных системах на основе графов потока вызовов

Данила Пучкин

Разработчик-исследователь

Группа Системных Исследований
Департамент Перспективных Технологий

kaspersky

Agenda

Мониторинг потока управления

Сигнатура потока управления

Обход сигнатуры потока управления

Граф путей потока управления

Agenda

Мониторинг потока управления

Графы потока вызовов

Обход сигнатуры потока управления

Граф путей потока управления

Исправность исполнения процесса

Функционирование процесса в соответствии с предъявляемыми ему требованиями

Исправность объекта
Состояние объекта, при котором его характеристики функционирования соответствуют предъявленным ему требованиям

Надежность (достоверность, reliability) объекта
Свойство соответствия объекта предусмотренному поведению и результатам

Надежность объекта – один из критериев исправности объекта

Мониторинг потока управления

Сравнение фактического потока управления процесса с некоторой эталонной моделью

Нарушение потока управления процесса – один из признаков нарушения исправности процесса

Мониторинг потока управления может быть использован для контроля исправности процесса

При выявлении несоответствия исполняемого потока управления своему эталону поток управления считается нарушенным

Мониторинг потока управления на основе целевых команд

Целевые команды (инструкции)
Команды, информация об исполнении процессом которых доступна от операционной системы, в рамках которой функционирует процесс

Сигнатура потока управления
Некоторая модель потока управления, задающая последовательности целевых команд, которые могут быть исполнены исправным процессом

Патент «Система и способ контроля работоспособности процессов в операционной системе» / Сорокин И.А., Пучкин Д.А., Духвалов А.П. ; заявитель АО «Лаборатория Касперского» (RU)

Agenda

Мониторинг потока управления

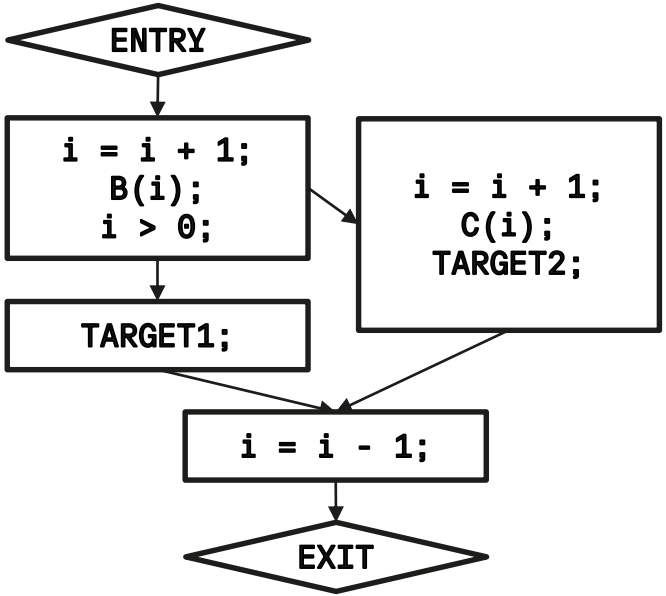
Сигнатура потока управления

Обход сигнатуры потока управления

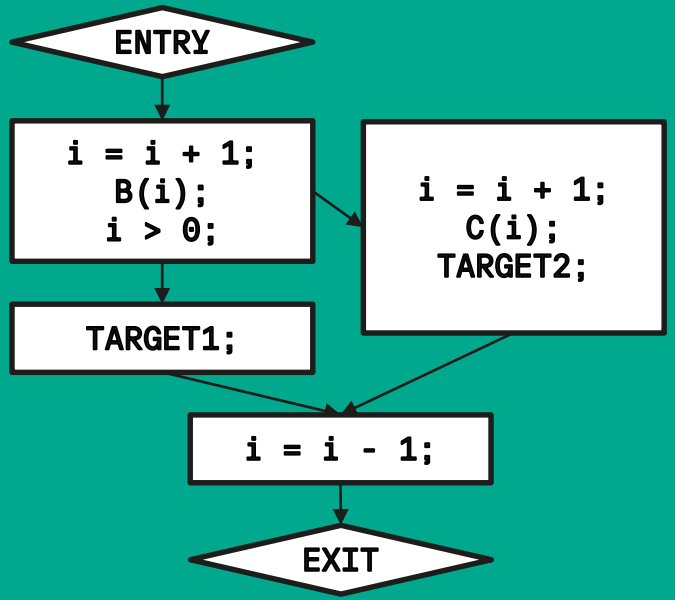
Граф путей потока управления

Граф потока управления

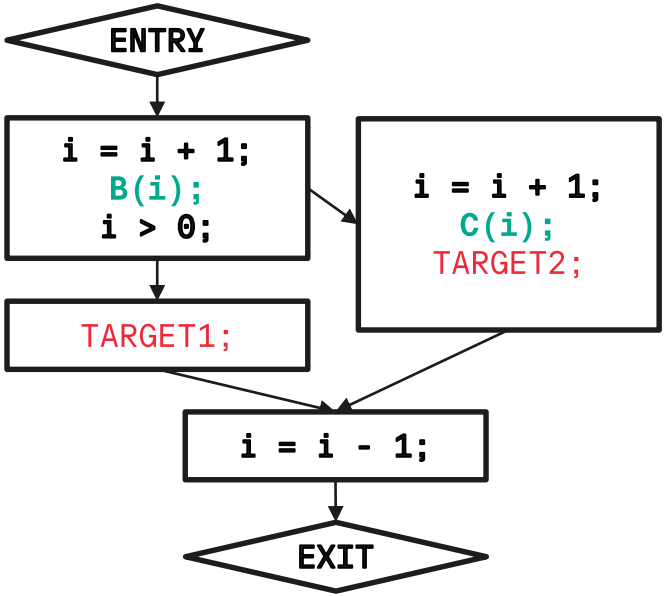

```
1  void A(int i) {
2      B (i++);
3      if (i > 0) {
4          TARGET1;
5      } else {
6          C (i++);
7          TARGET2;
8      }
9      i--;
10 }
```



Выделение целевых и управляющих команд

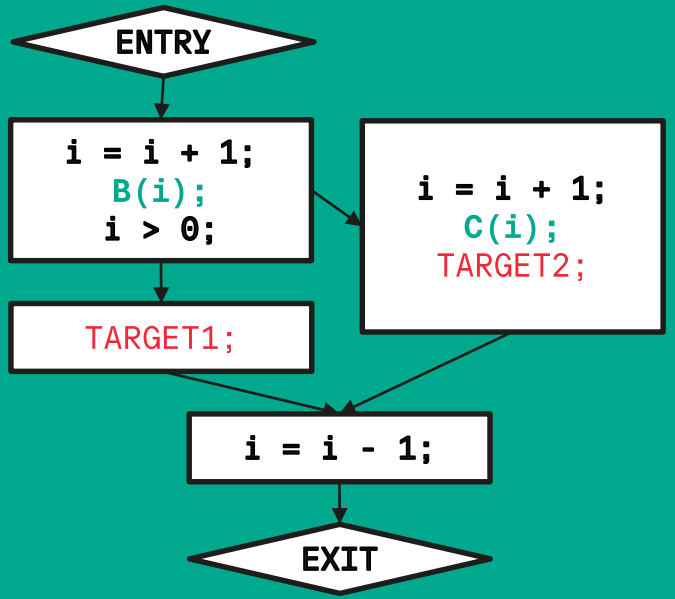


Целевые команды
Команды, информация об исполнении процессом которых доступна от операционной системы

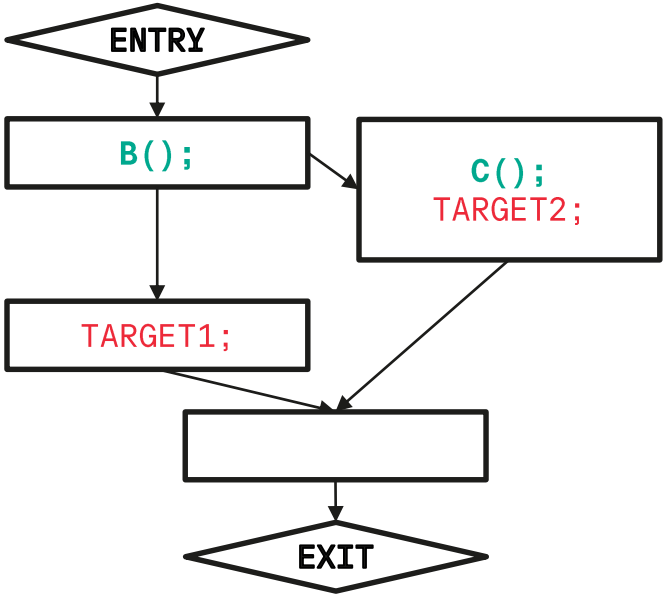


Управляющие команды
Команды, определяющие поток управления: вызовы функций, команды перехода, команды создания потоков

Удаление команд

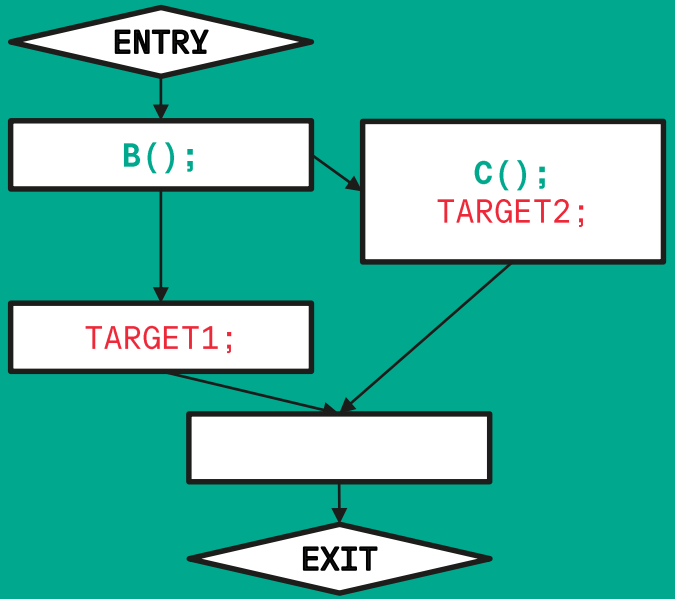


Из вершин удаляются инструкции, которые не являются целевыми или управляющими инструкциями

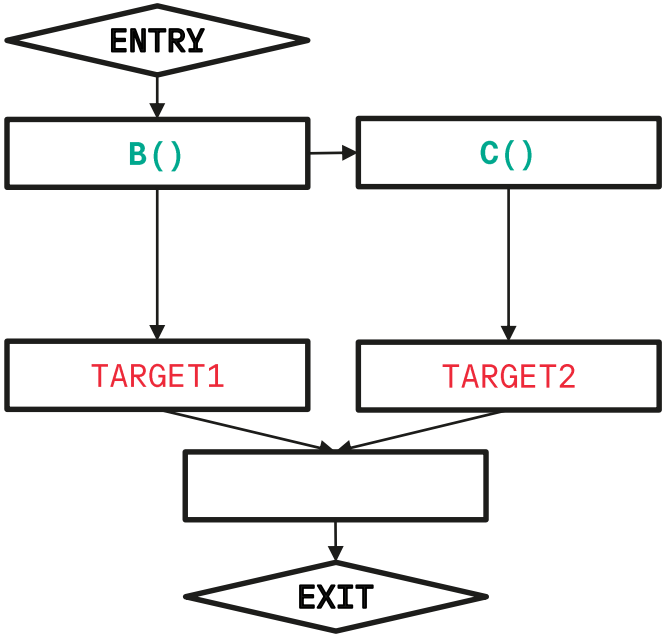


Если в вершине нет целевых или управляющих команд, из вершины удаляется вся информация о командах

Замена вершин на пути

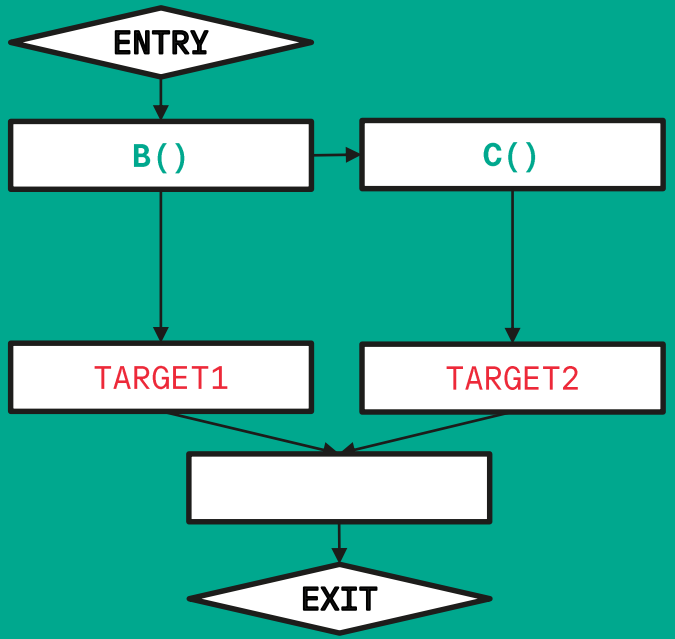


Каждая вершина, содержащая более одной инструкции, заменяется на путь

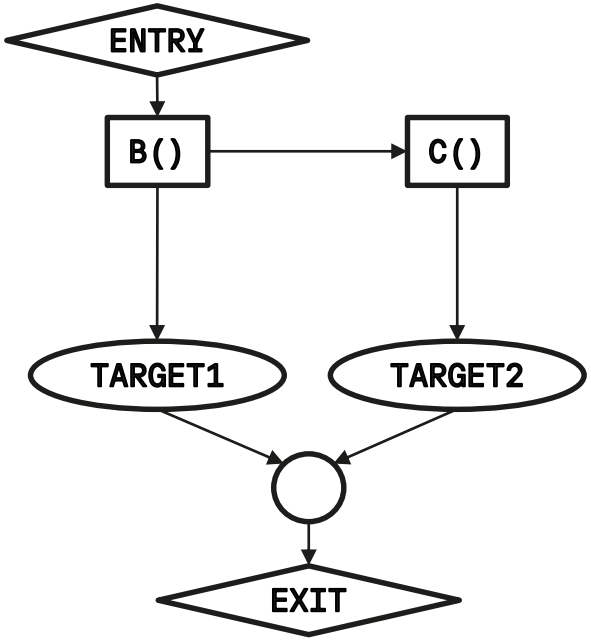


Если вершине соответствует одна или ни одной команды, ее не требуется преобразовывать

Типизация вершин

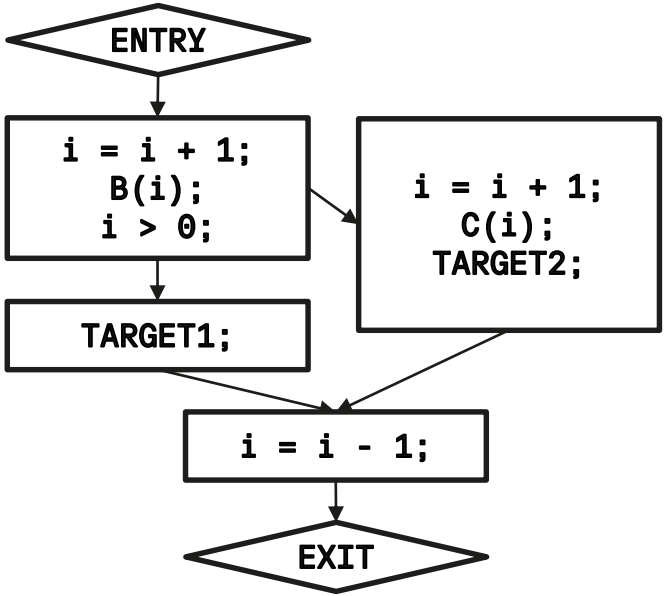


Каждой вершине соответствует или одна команда, или ни одной команды

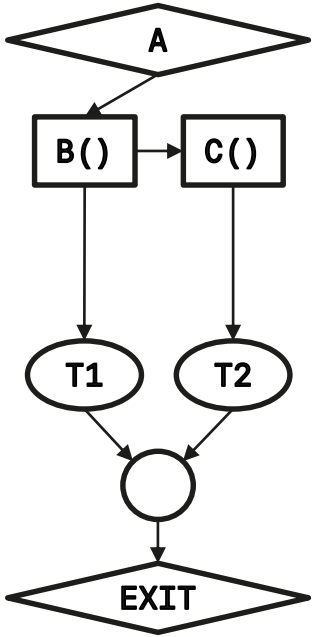


Целевые вершины
Управляющие вершины
Пустые вершины

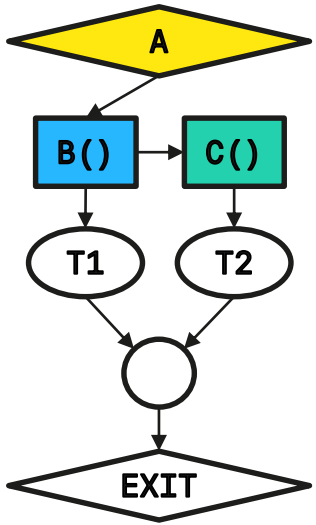
Граф потока вызовов



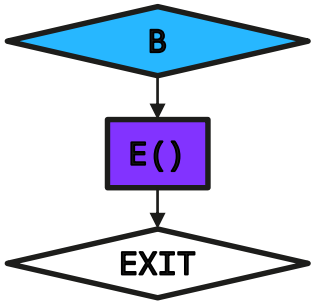

```
1  void A(int i) {
2      B (i++);
3      if (i > 0) {
4          TARGET1;
5      } else {
6          C (i++);
7          TARGET2;
8      }
9      i--;
10 }
```



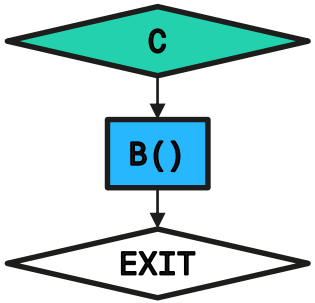
Связывание графов потока вызовов



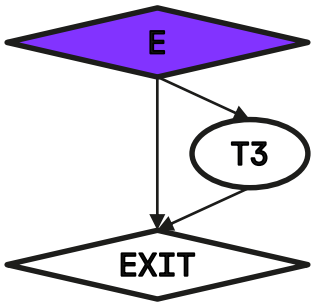

```
1 void B(int i) {
2     E(i--);
3 }
```



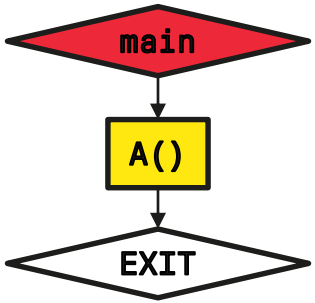

```
1 void C(int i) {
2     B(i++);
3 }
```




```
1 void E(int i) {
2     if (i > 0) {
3         TARGET3;
4     }
5 }
```




```
1 int main() {
2     A(0);
3 }
```



Pointer Analysis / Yannis Smaragdakis, George Balatsouras – 2015.

Agenda

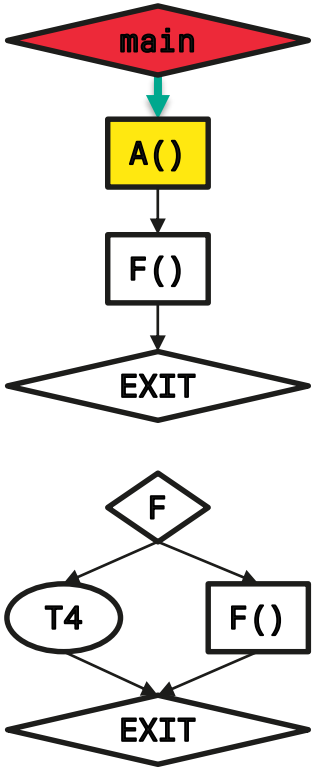
Мониторинг потока управления

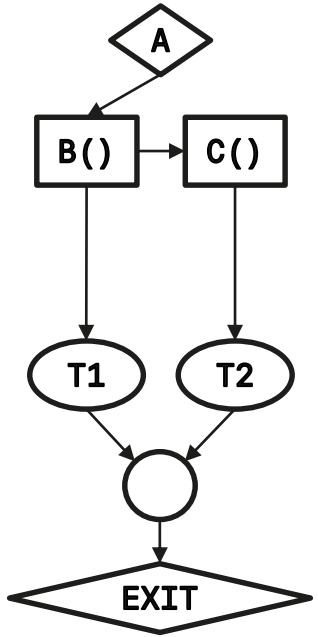
Сигнатура потока управления

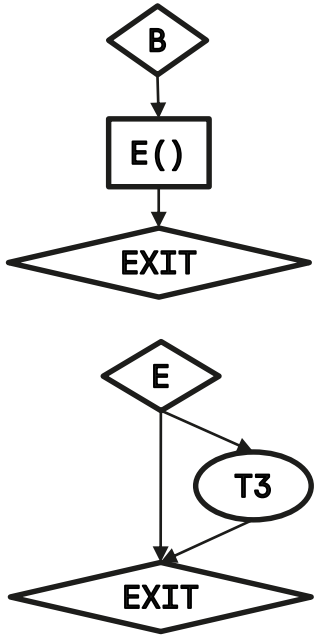
Обход сигнатуры потока управления

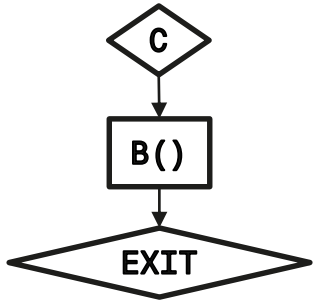
Граф путей потока управления

Проход по графу функции





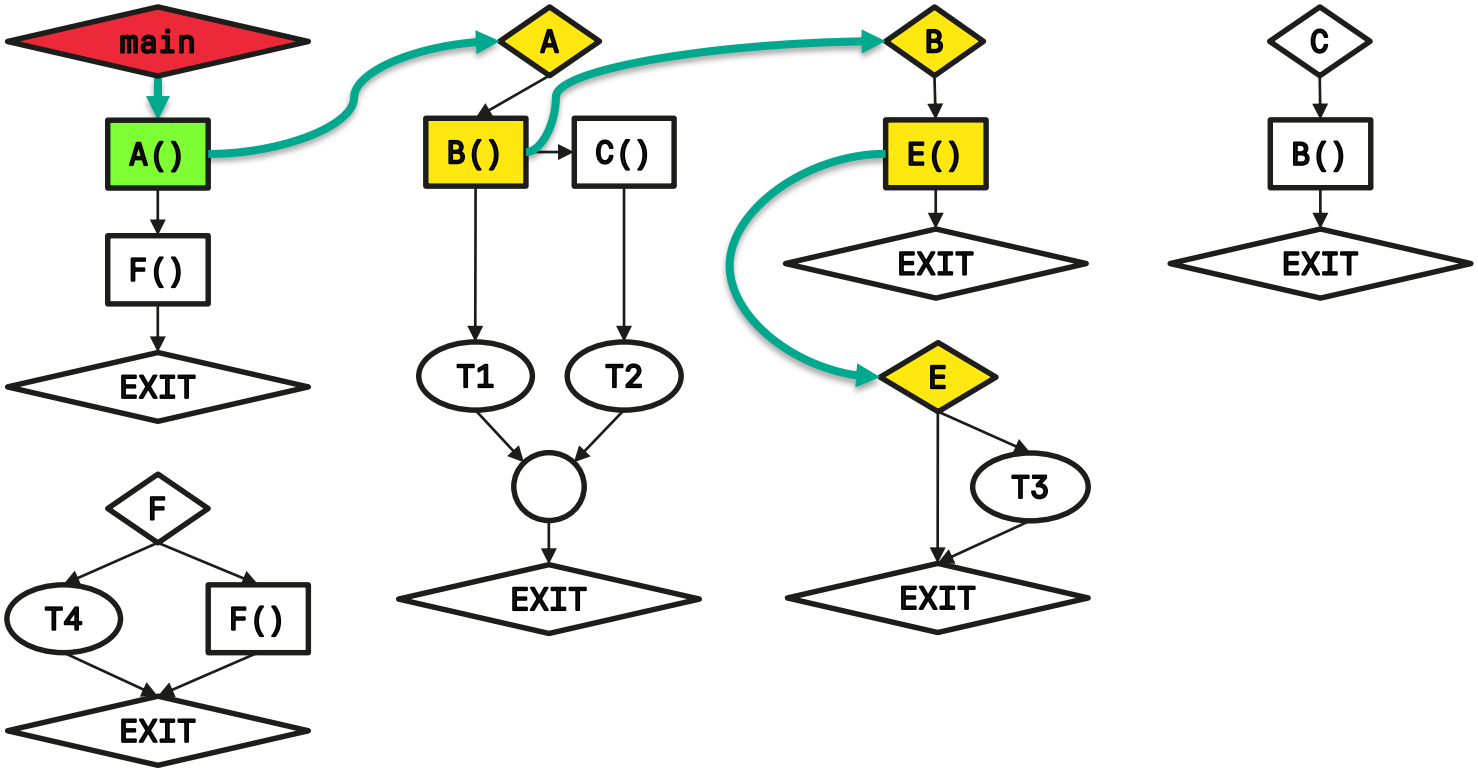




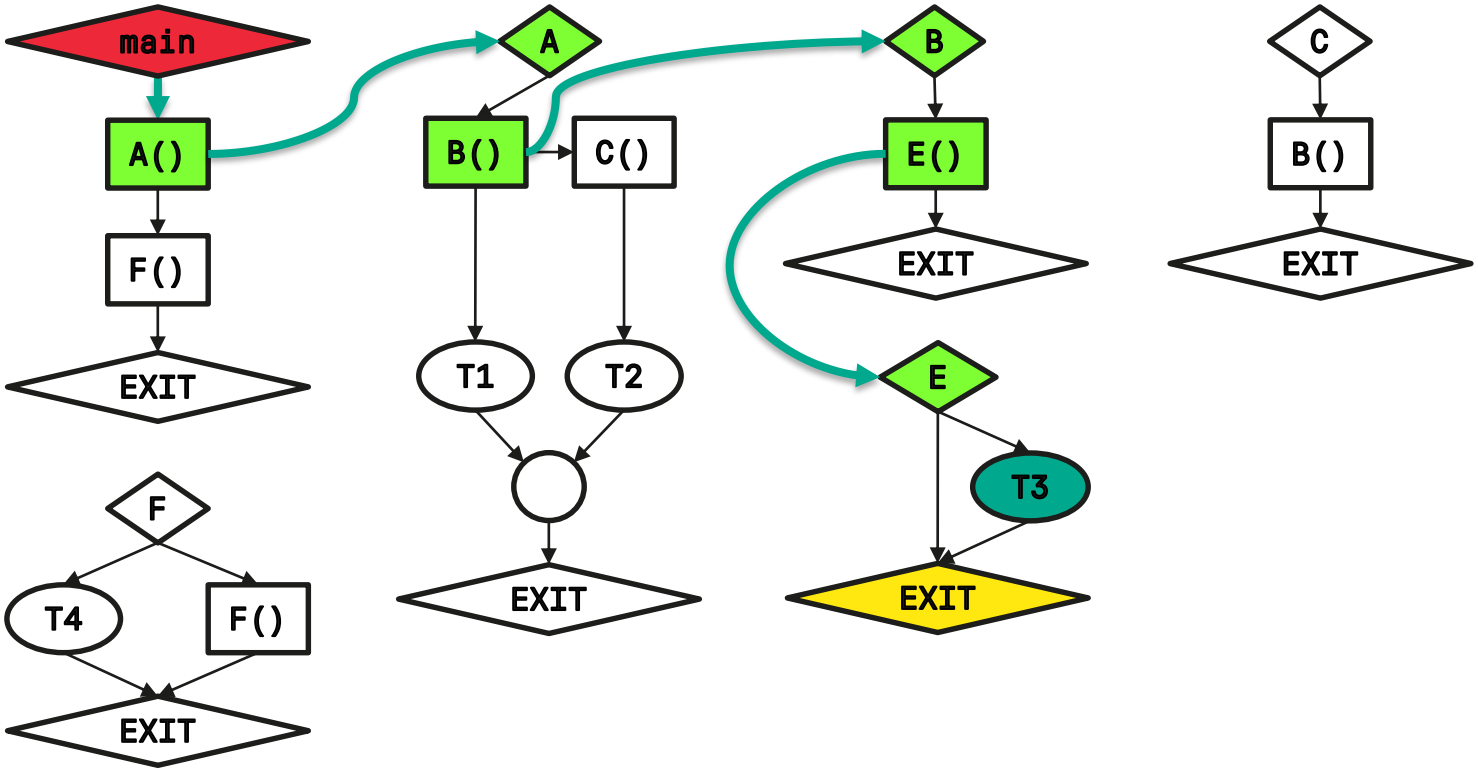
Шаг алгоритма мониторинга

1. Получение исполненной процессом команды
2. Валидация исполненной команды с допустимыми
3. Обход сигнатуры с целью получения следующих допустимых целевых команд

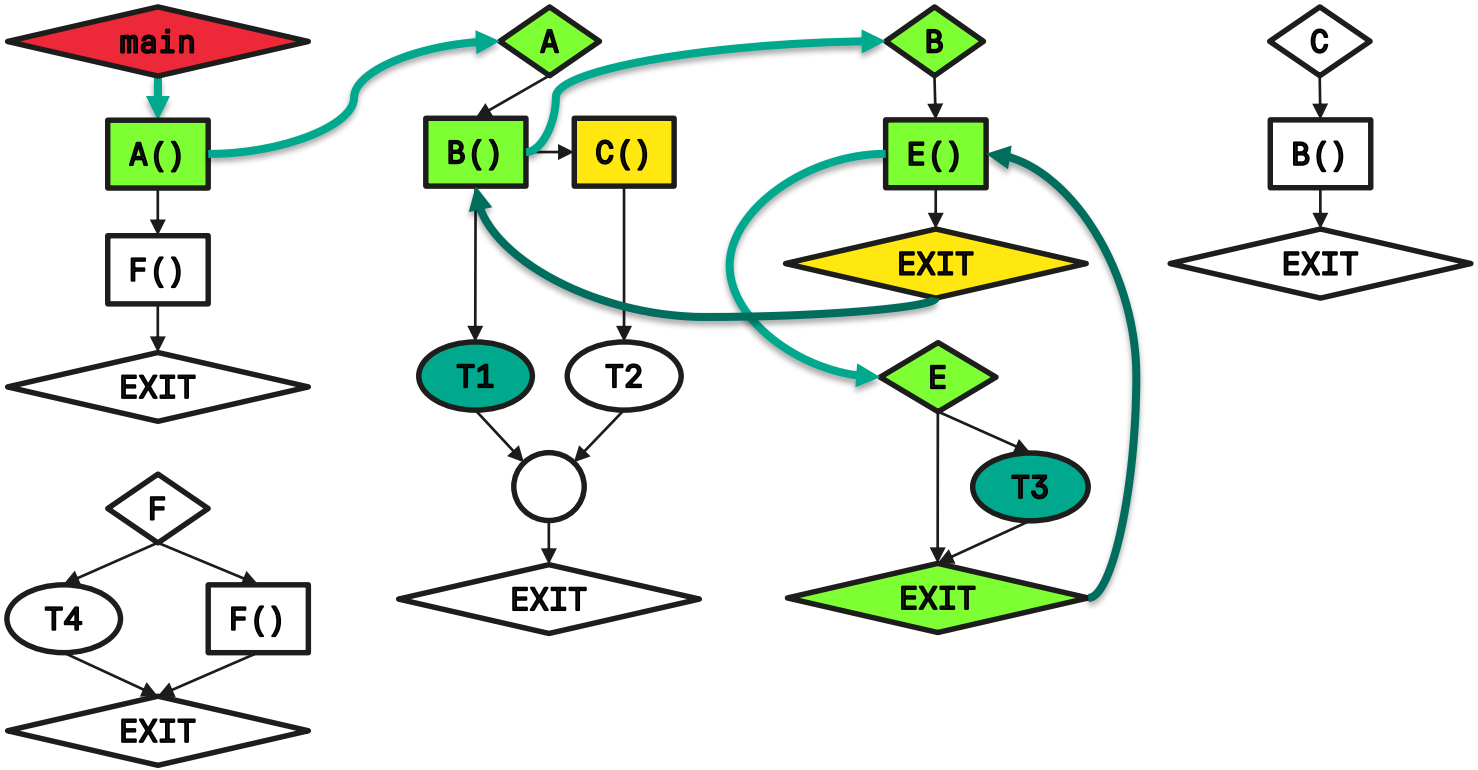
Вход в граф функции



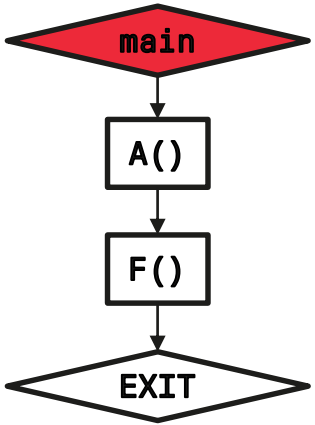
Ветвления потока управления

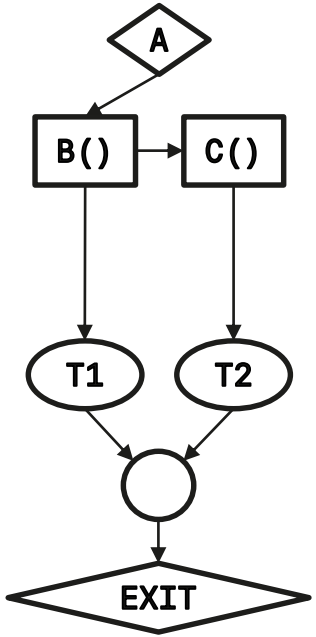


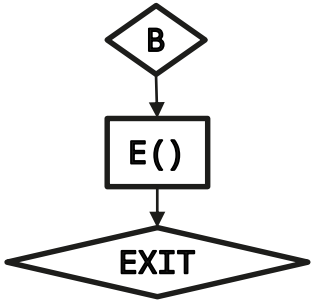
Выход из графа функции

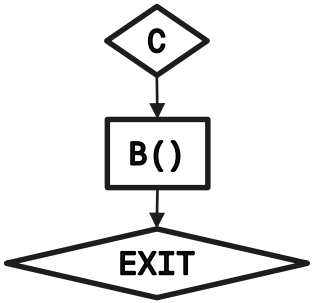


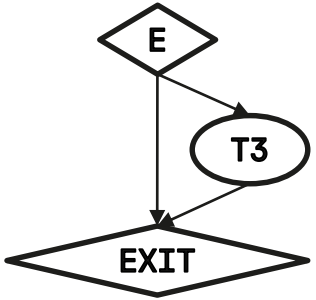
Проблемы обхода сигнатуры

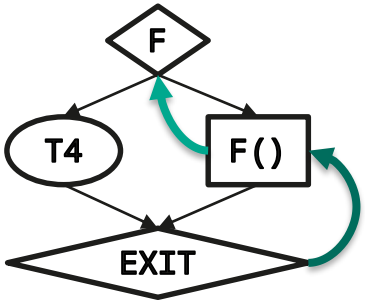


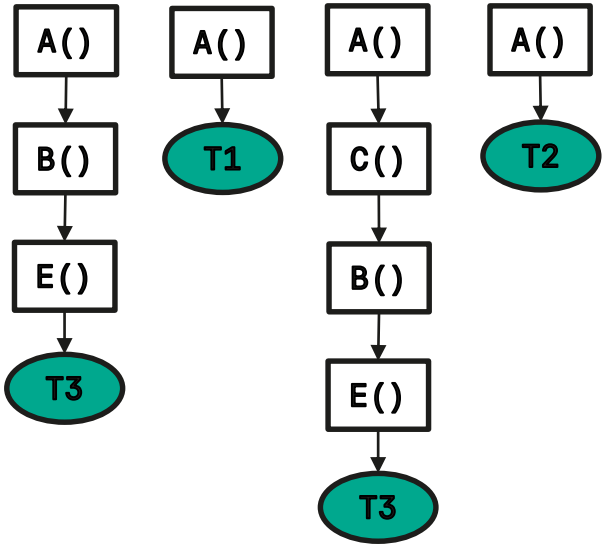












Agenda

Мониторинг потока управления

Сигнатура потока управления

Обход сигнатуры потока управления

Граф путей потока управления

Граф путей потока управления

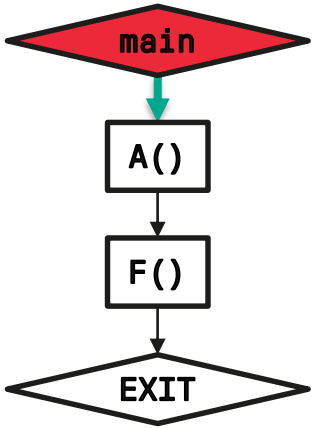
Граф путей
Ориентированный граф, каждая вершина которого соответствует единственной вершине сигнатуры потока управления, а ребра отражают связи по управлению между ними

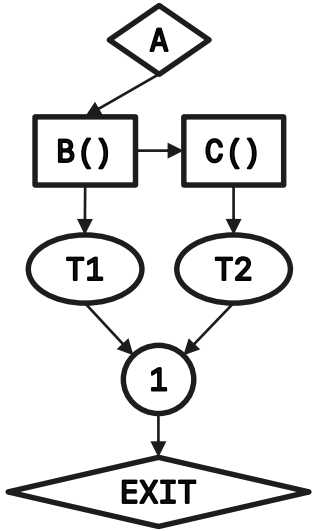
Пустой граф путей содержит корневую вершину, к которой привязывается стартовая вершина стартового графа функции

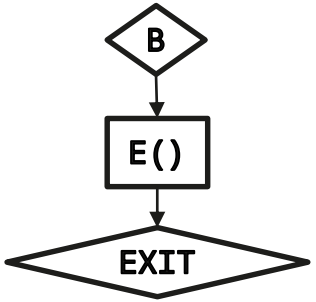
G.Pop (n)
Удаляет вершину n из графа G и возвращает ее предшественников

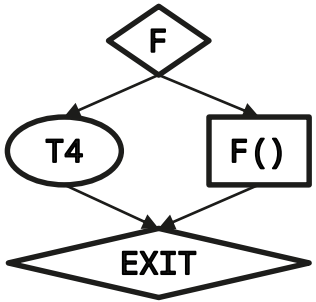
G.Append (n_from, n_to)
Вставляет вершину n_to и ребро (n_from, n_to) в граф G

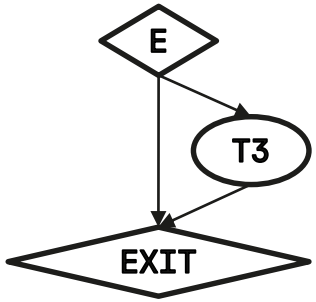
Проход по графу функции

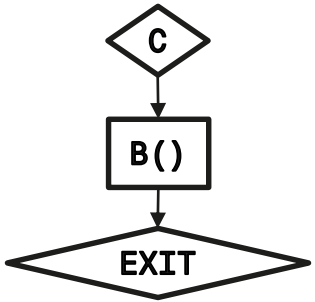


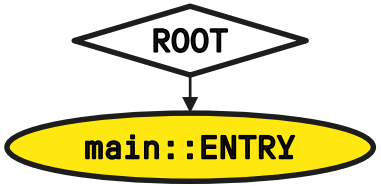




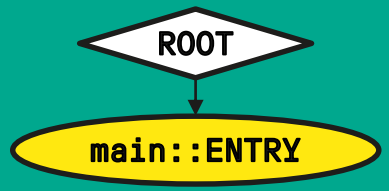


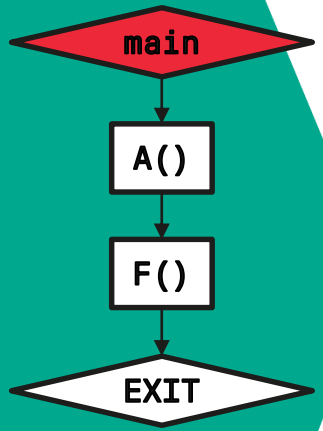


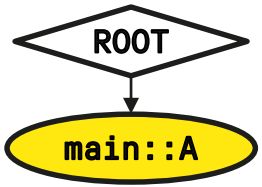




Through



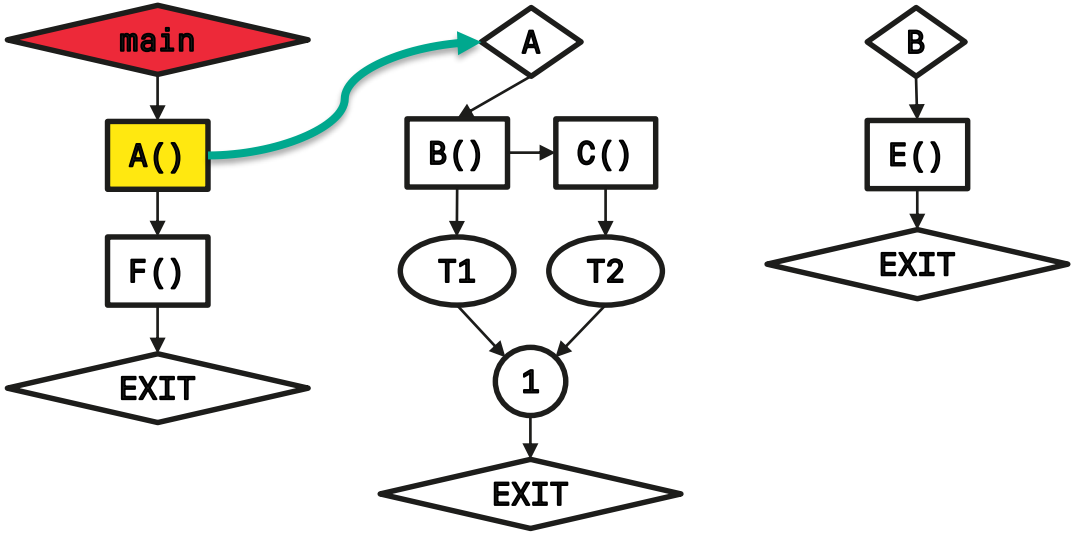


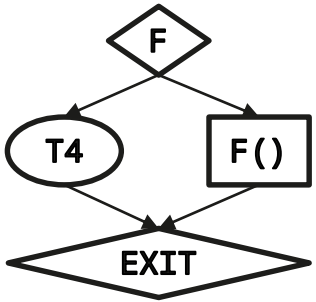


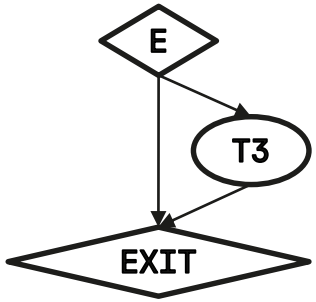
G.Through(n)
Заменяет вершину n на непустые вершины, полученные в результате обхода соответствующего графа функции

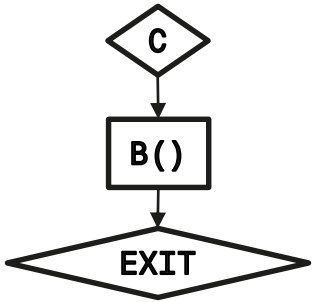
G.Through(main::ENTRY).

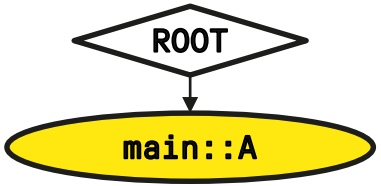
Вход в граф функции



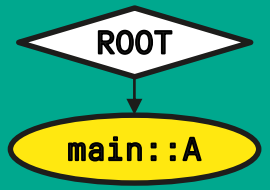


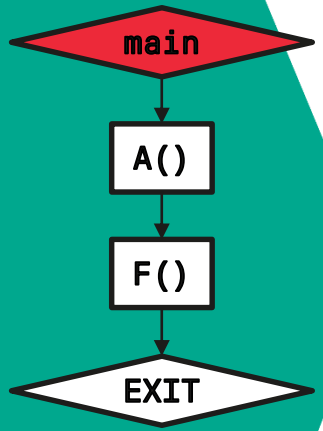


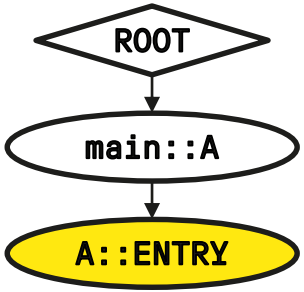




Into



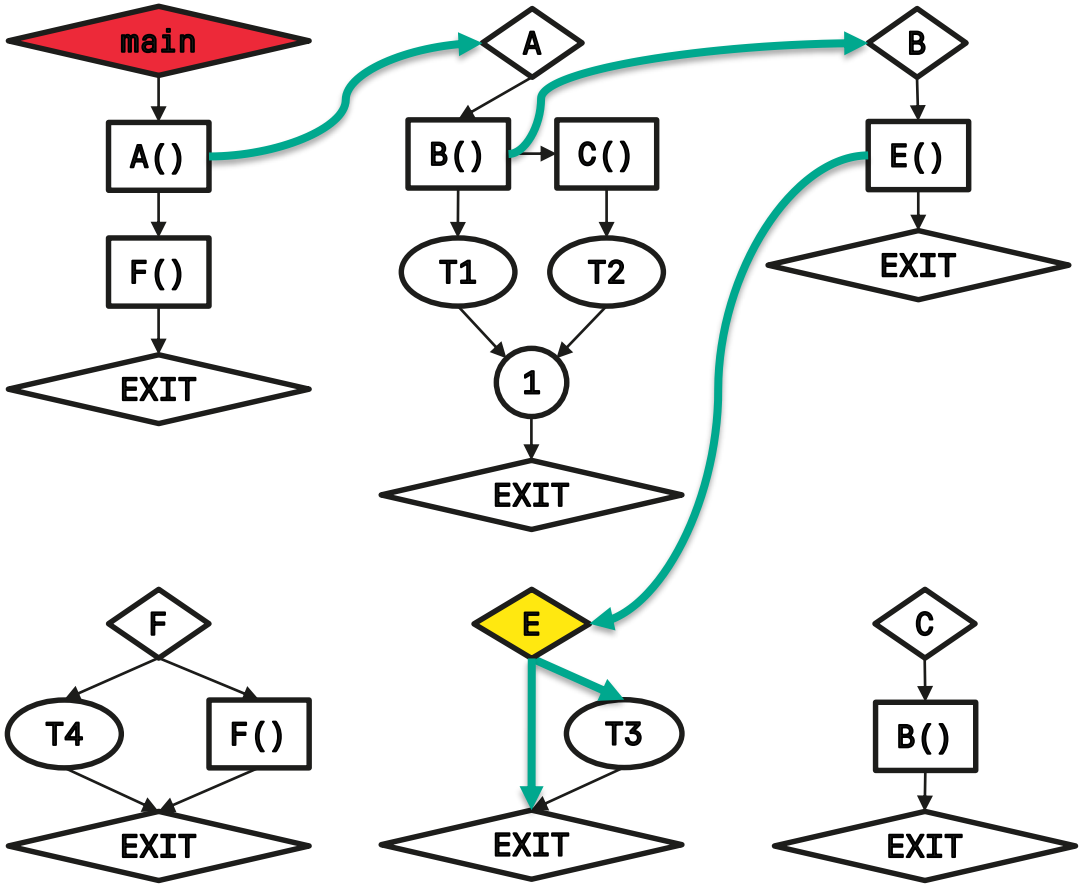


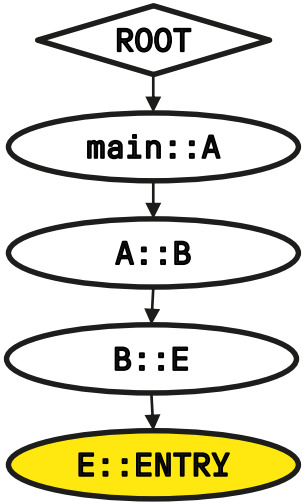


G.Into(n)
Присоединяет к вершине n стартовую вершину графа функции, которой вершина n соответствует

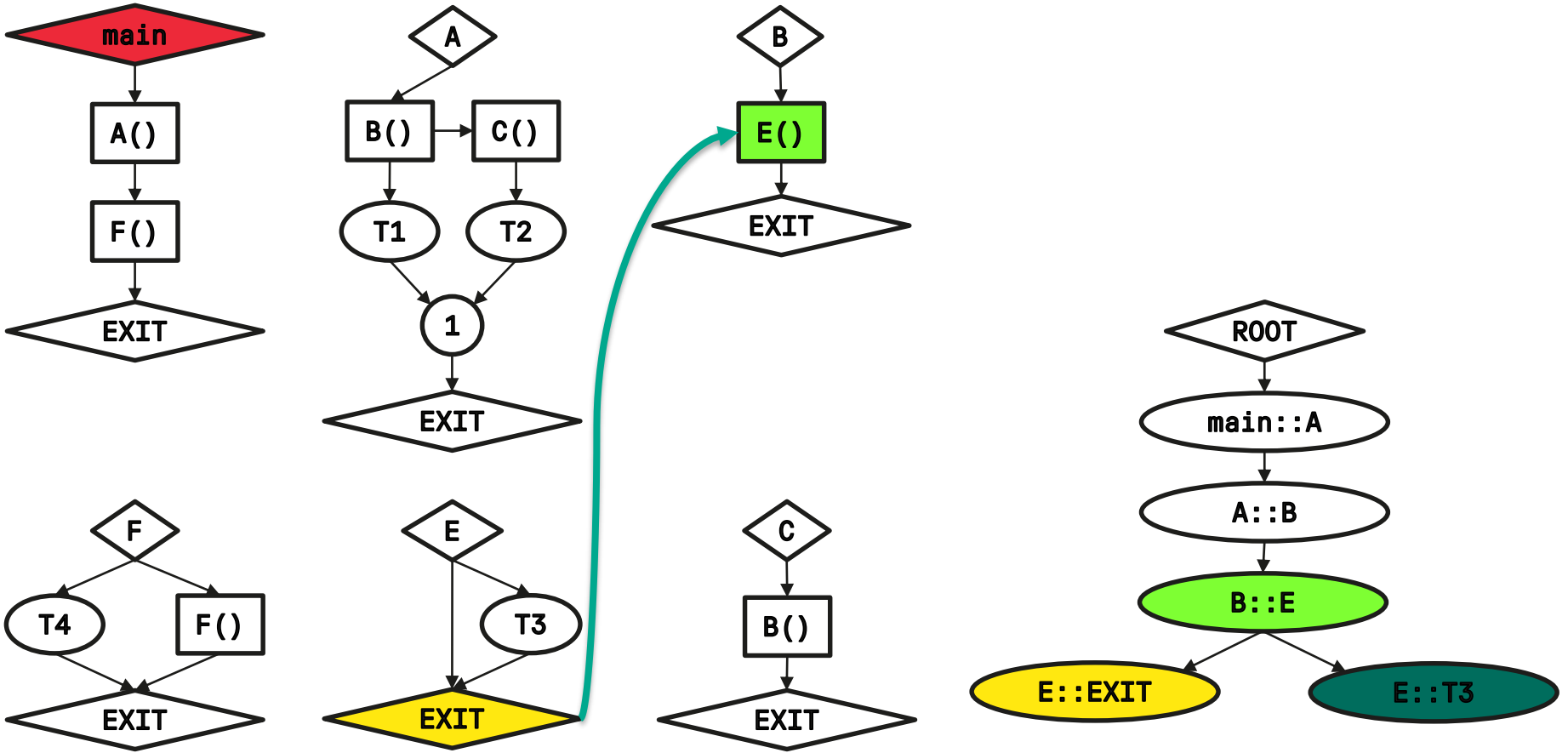
G.Into(main::A).

Проход по сигнатуре

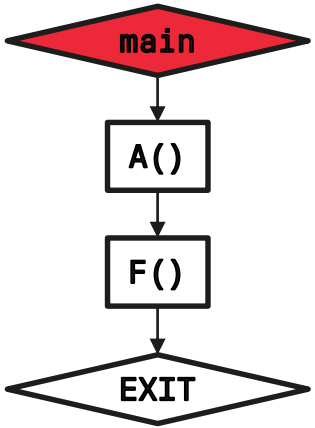


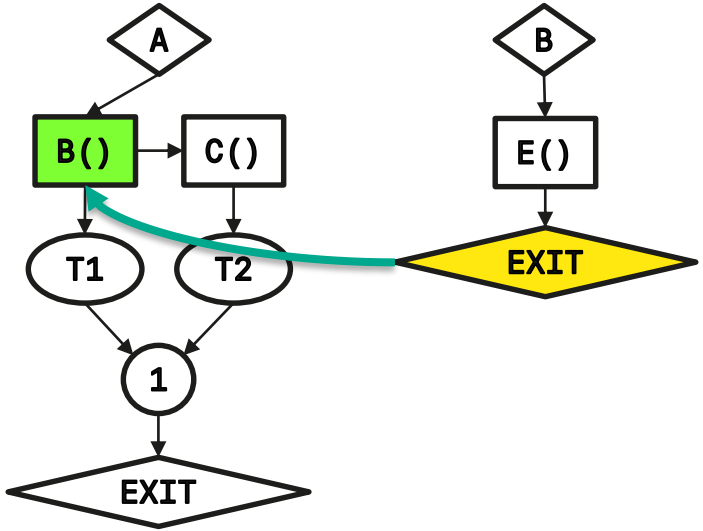


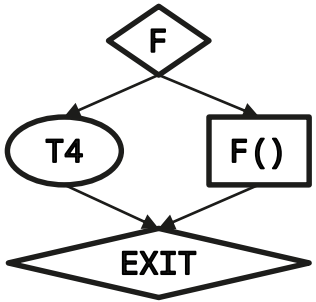
Выход из графа функции

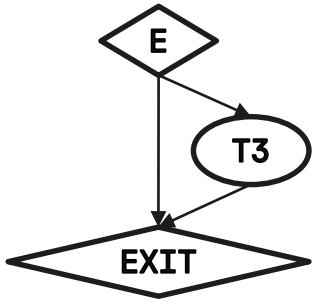


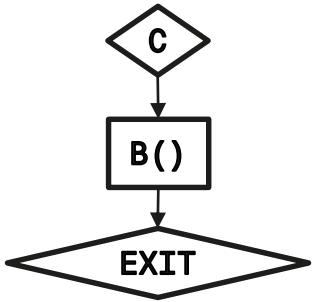
Выход из графа функции

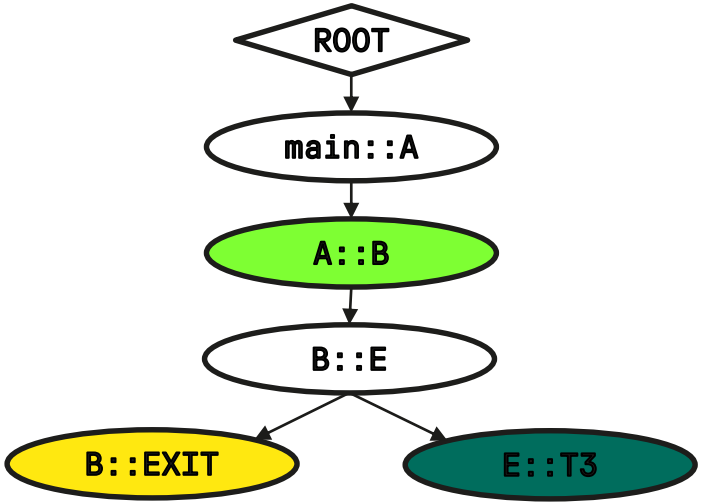


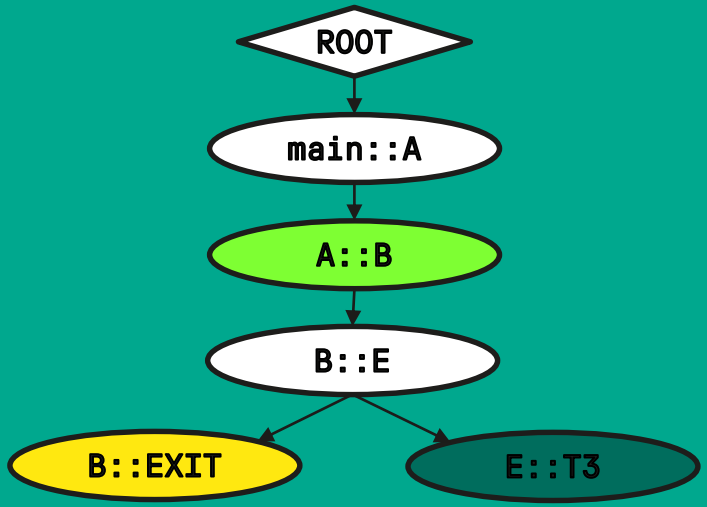


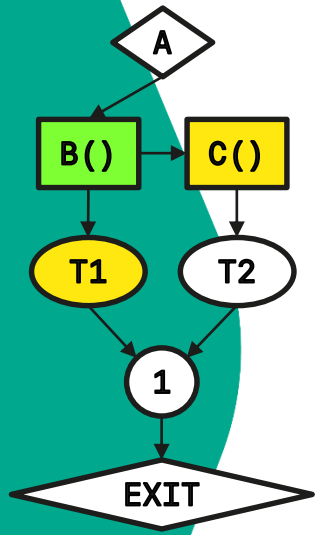




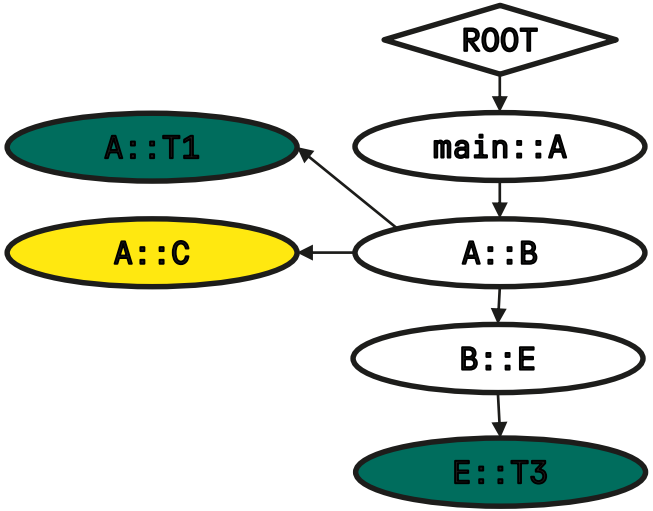






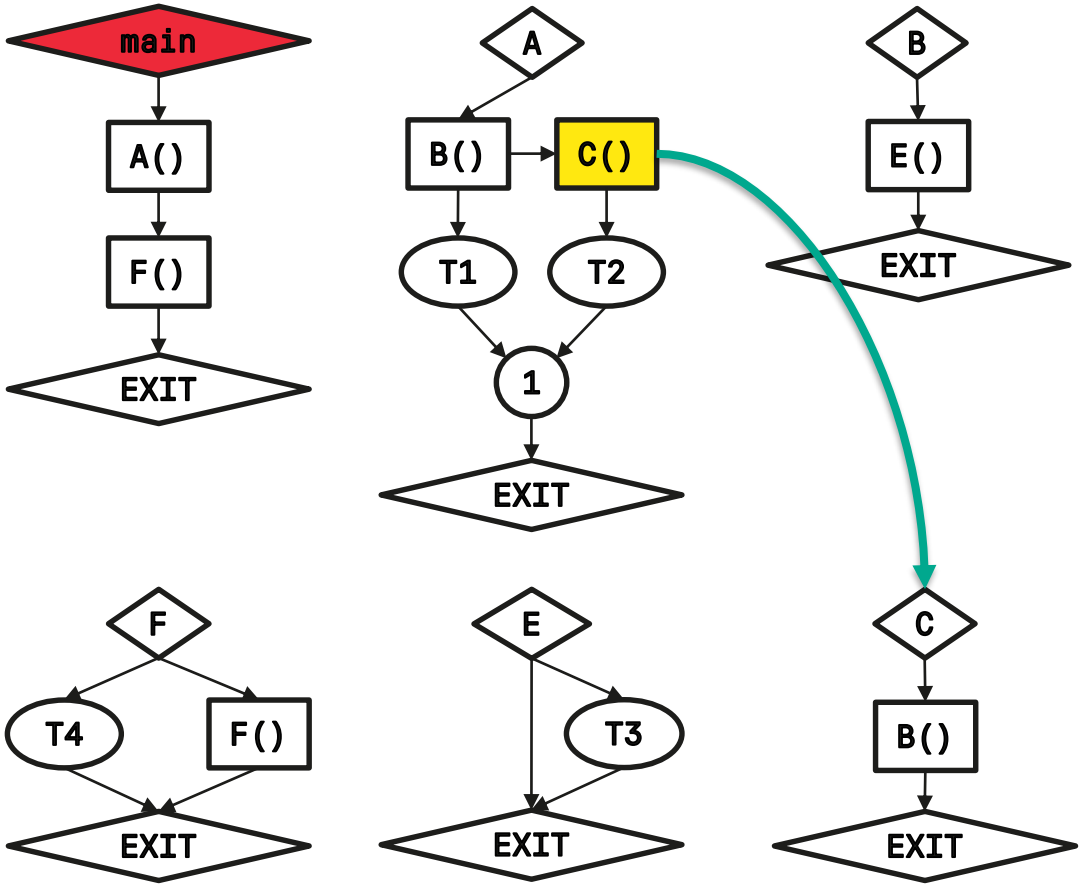


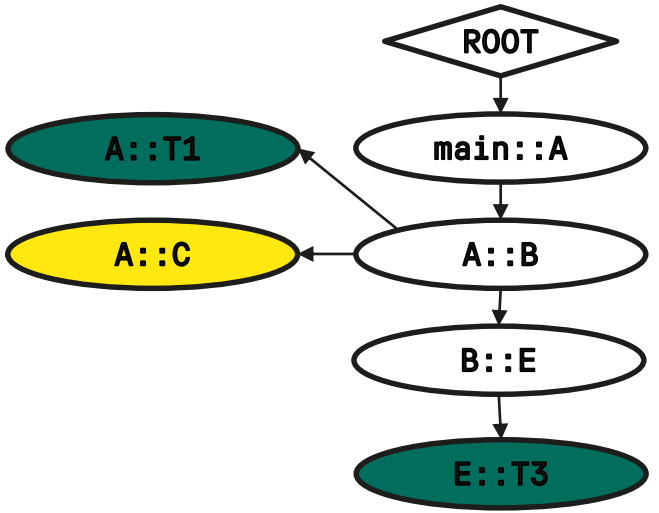
G.OutOf(n)
Определяет вершины возврата для выходной вершины n, после чего присоединяет к ним их последователей в своих графах функций



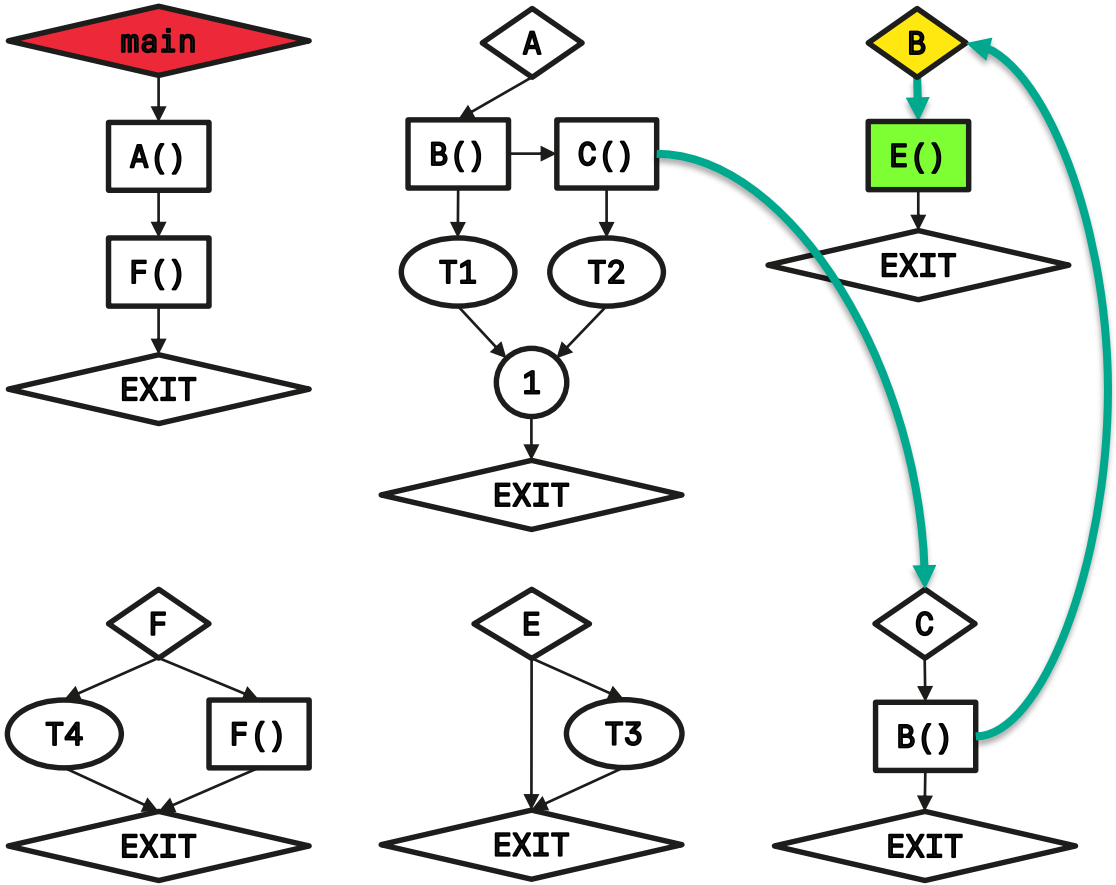
G. OutOf(B::EXIT).

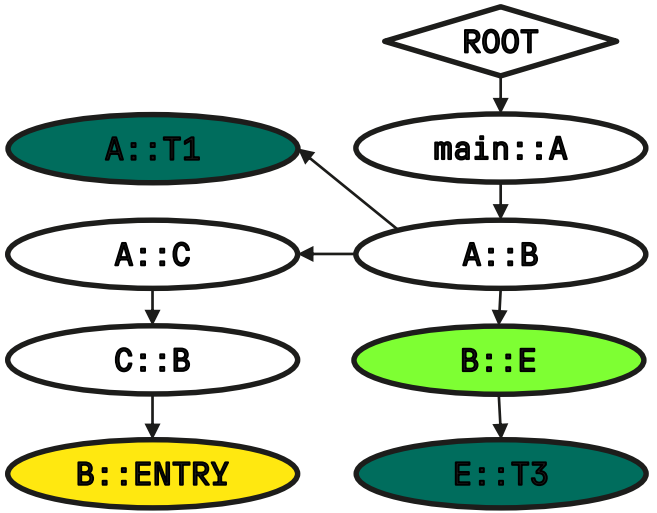
Выход из графа функции



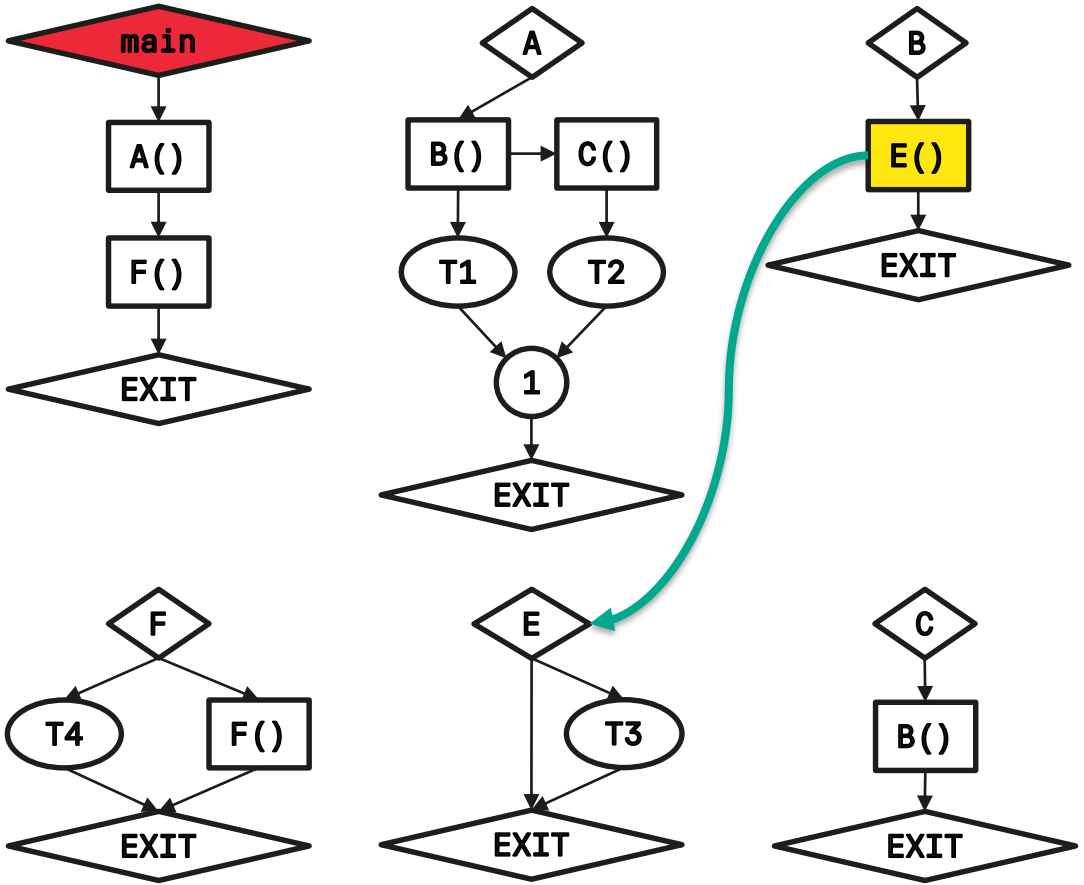


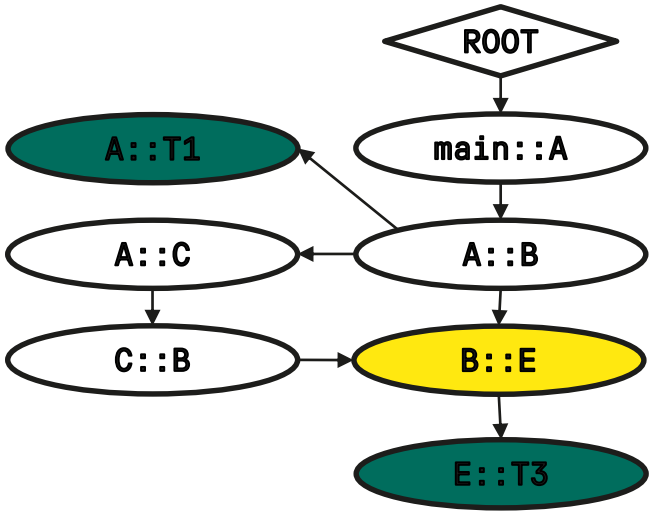
Повторный вход в граф функции



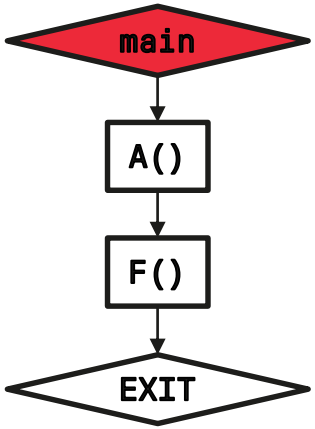


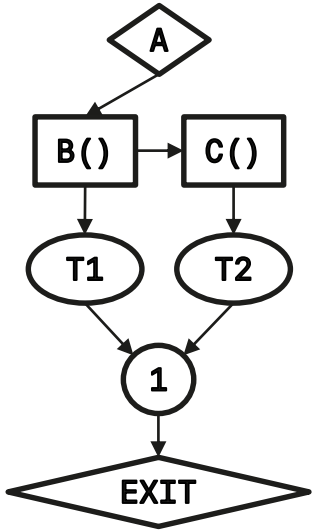
Повторный проход графа функции

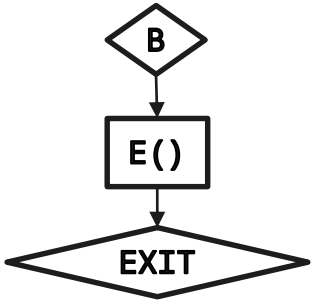


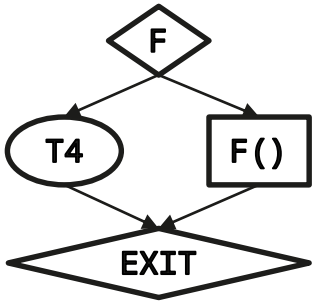


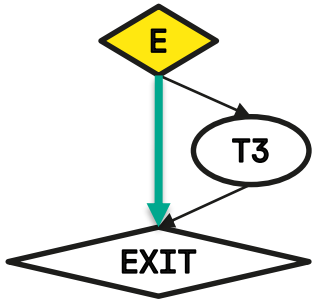
Повторный проход графа функции

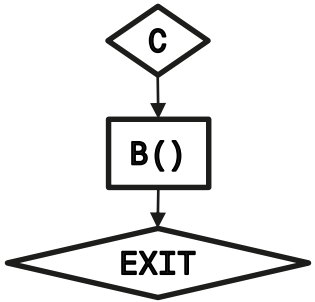


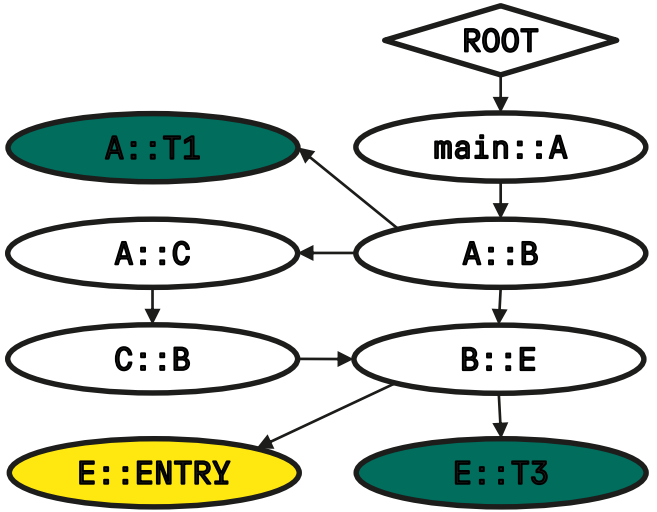




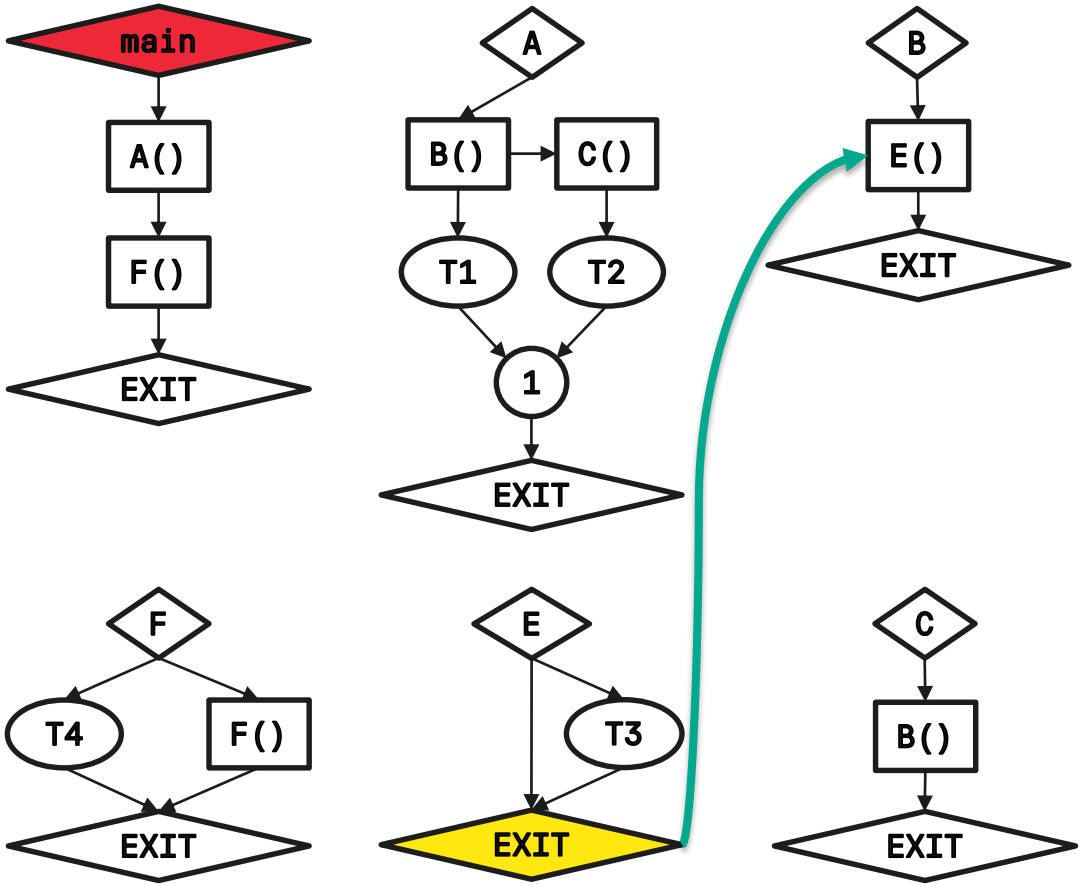


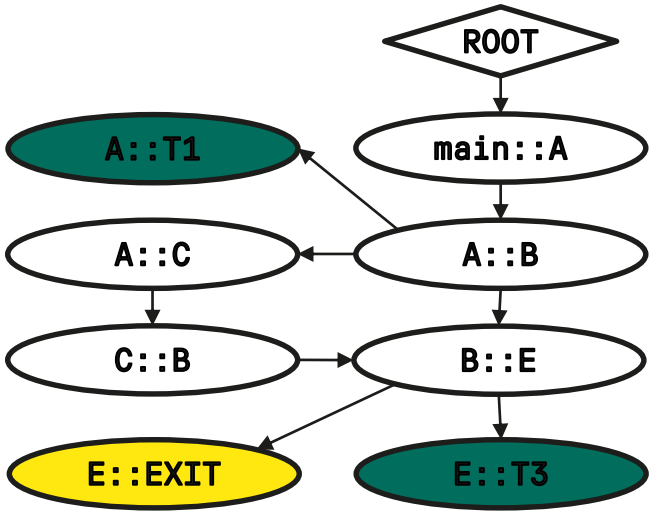




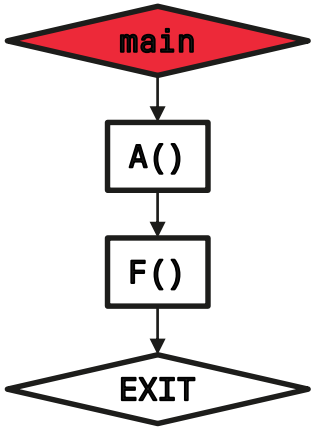


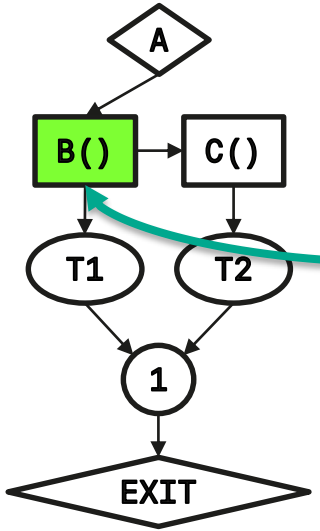
Повторный выход из графа функции

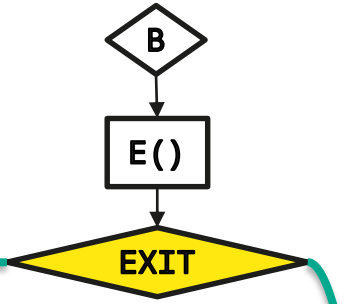


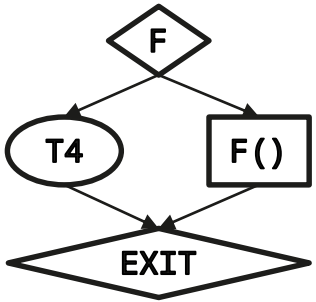


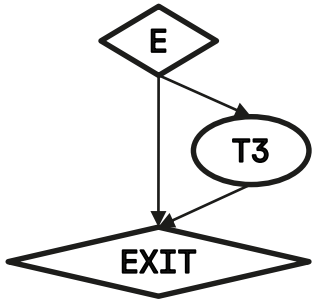
Повторный выход из графа функции

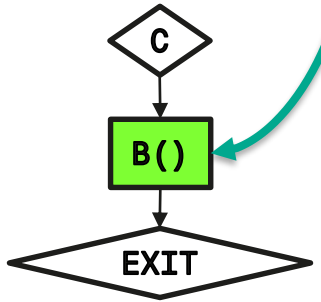


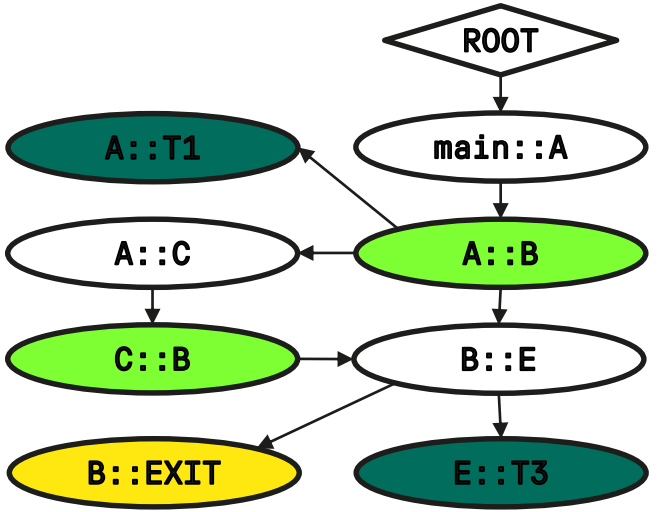




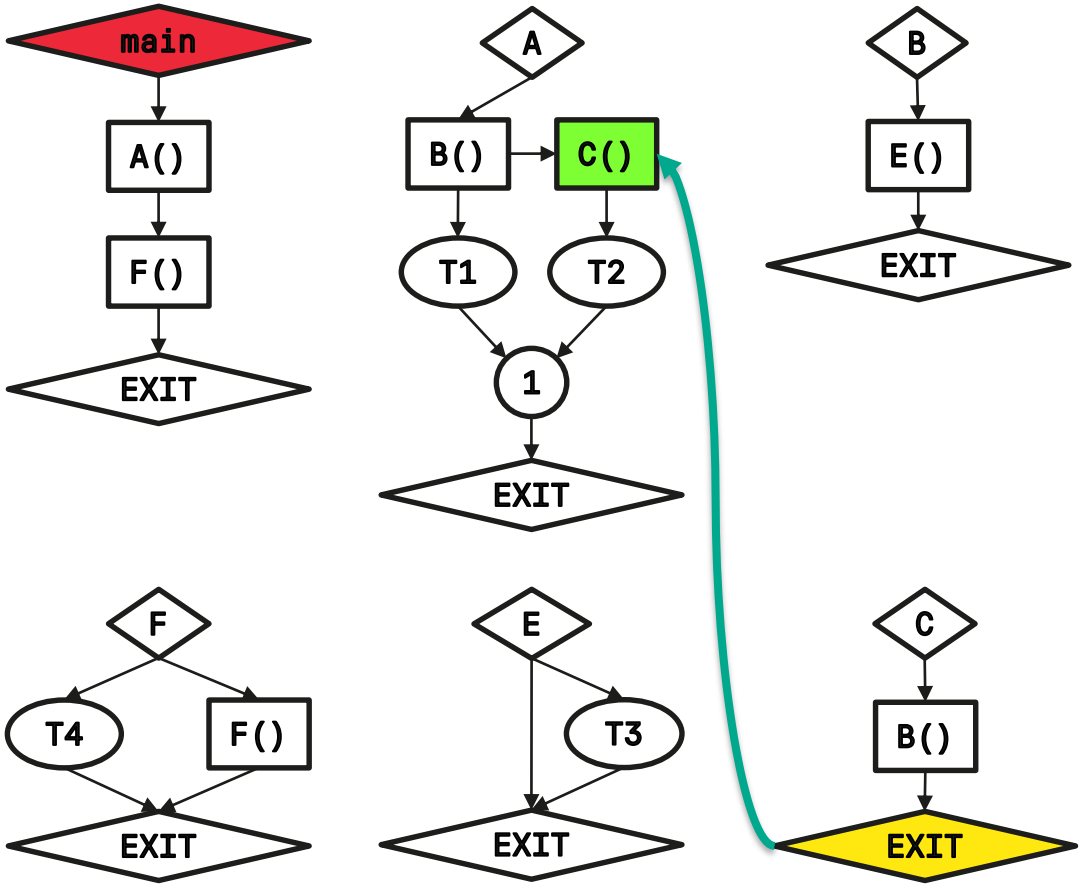


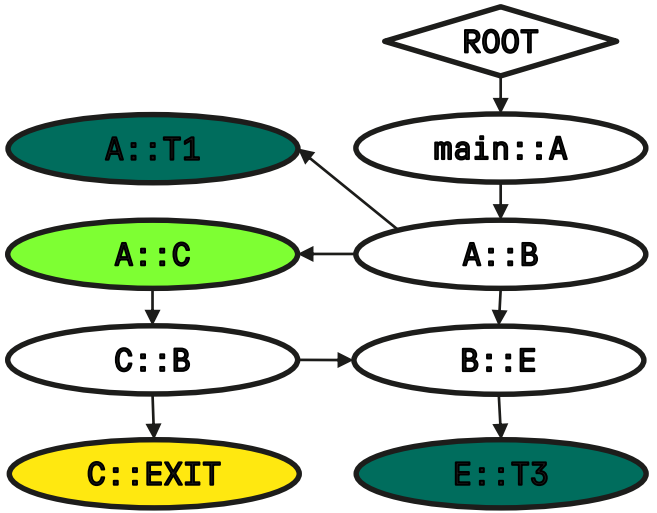




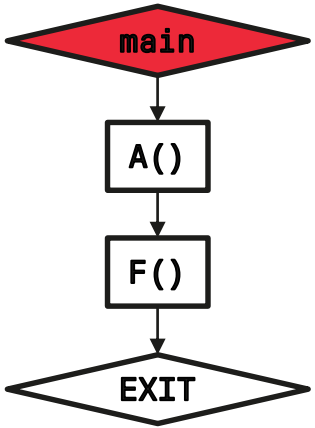


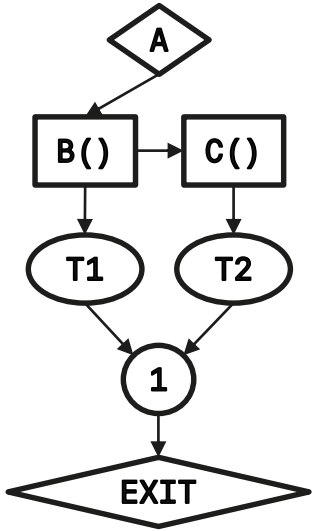
Повторный выход из графа функции

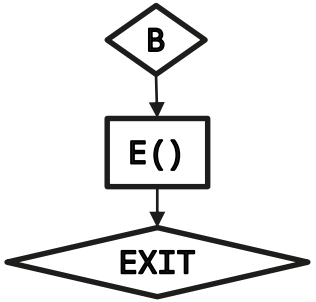


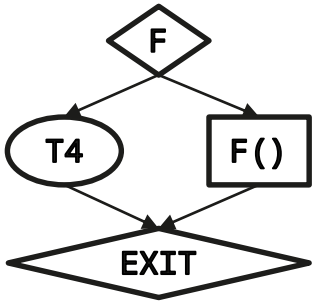


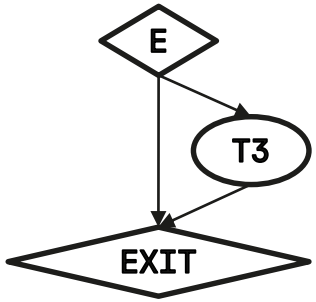
Результат обхода

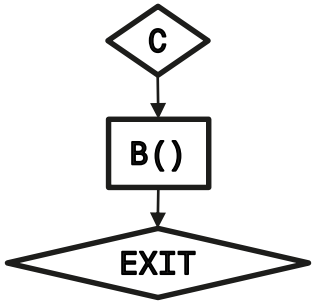


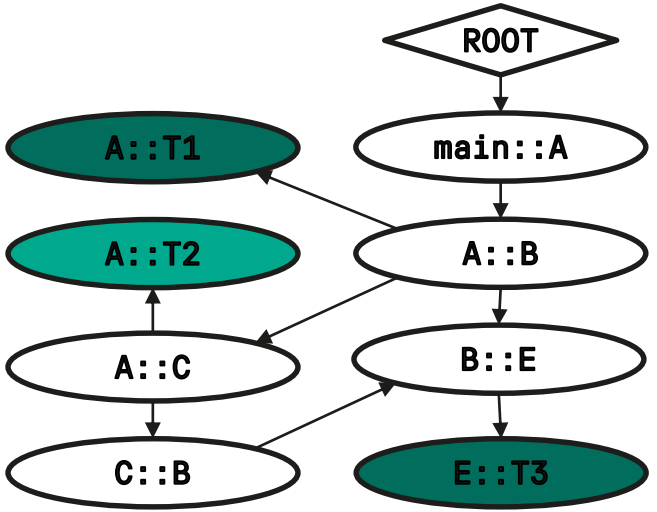




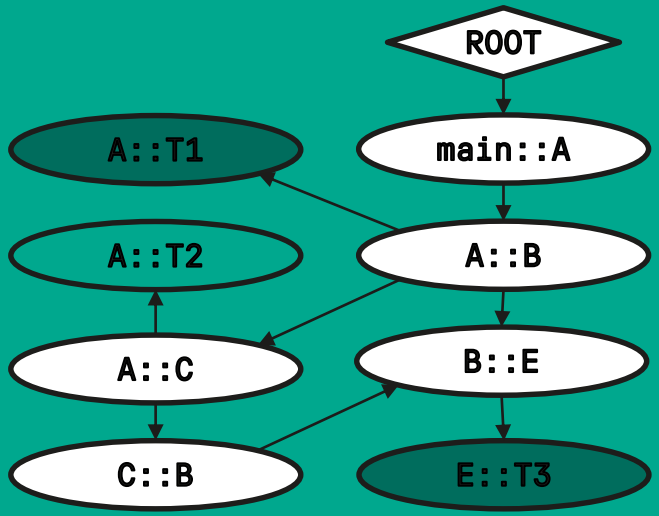




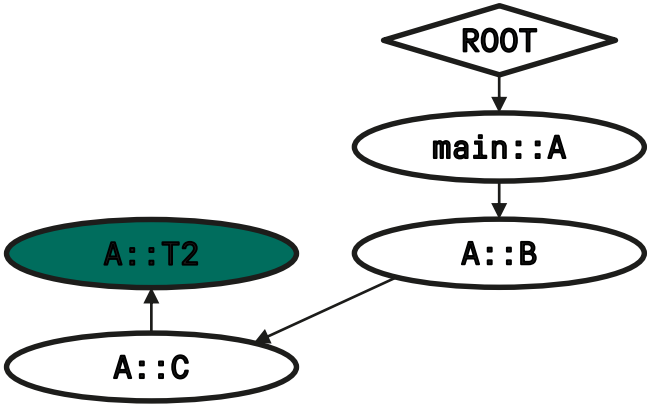




RemoveRedundant

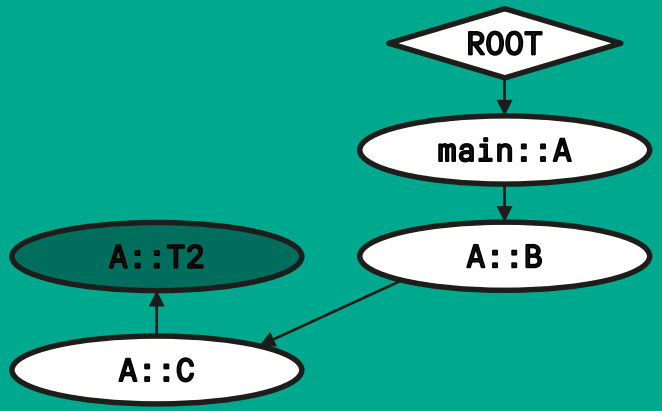


G.RemoveRedundant(T)
Удаляет из графа вершины, которые не принадлежат путям от корневой вершины до целевых вершин, соответствующих команде T

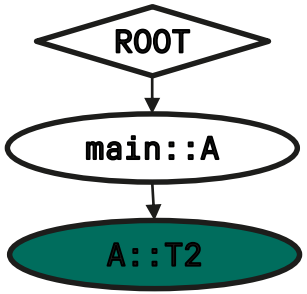


G.RemoveRedundant(T2)

Optimize

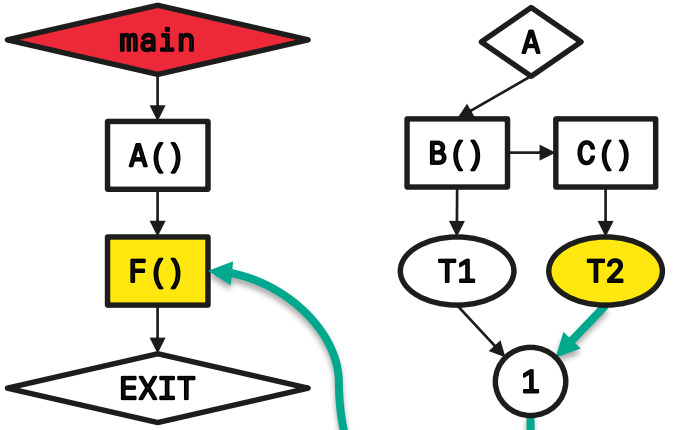


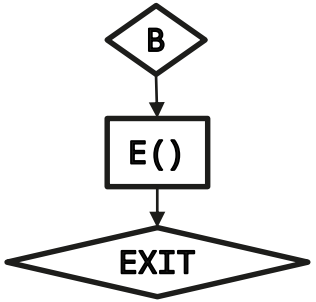
G.Optimize()
Удаляет из графа вершины, которые не несут информации о возврате потока управления

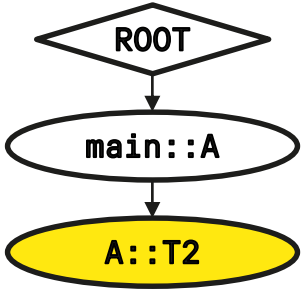


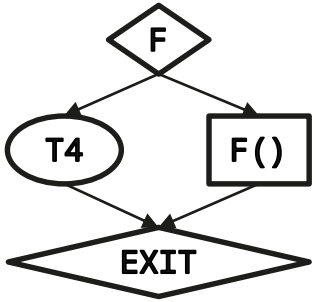
G.Optimize()

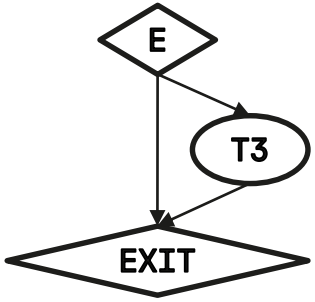
Обработка рекурсии

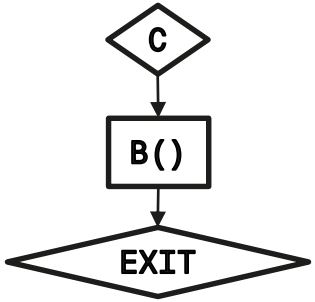




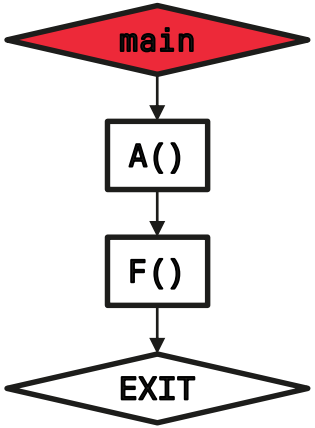


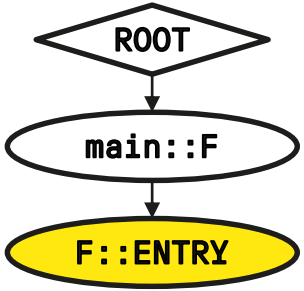


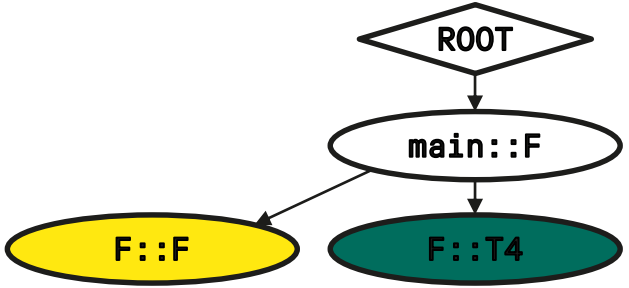


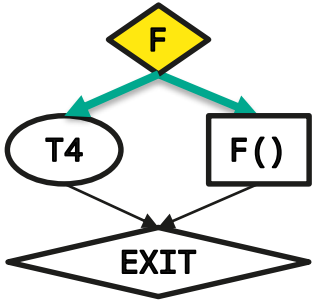


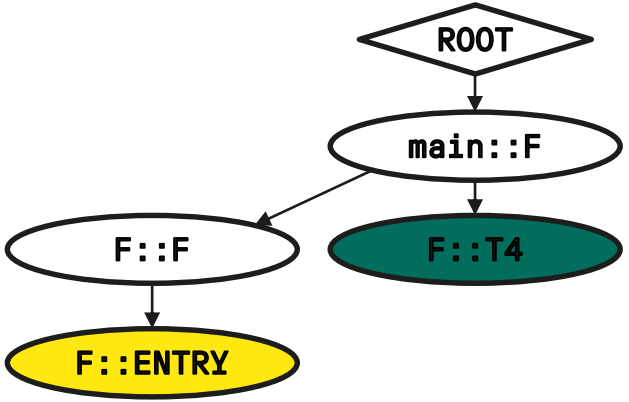
Обработка рекурсии



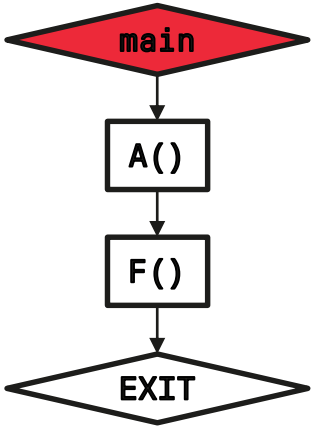


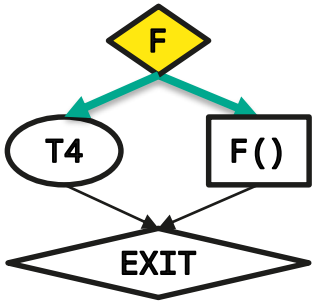


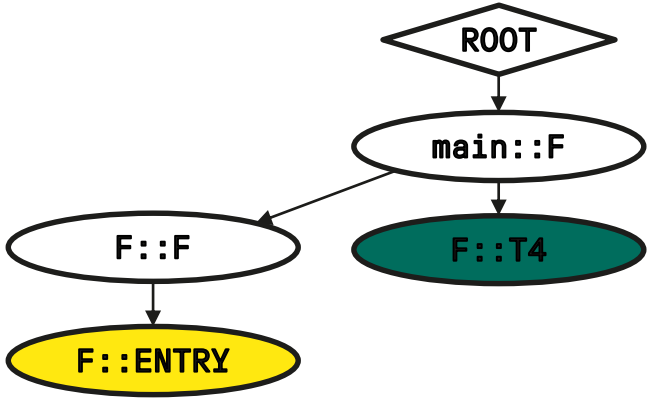


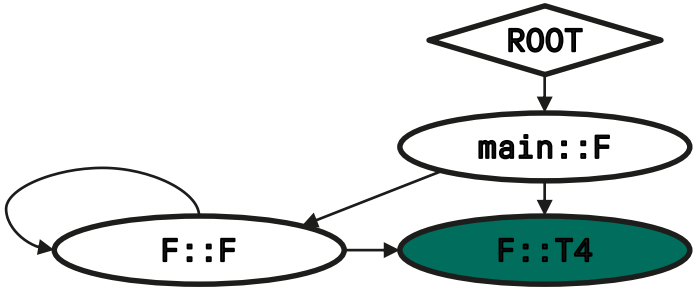


Обработка рекурсии

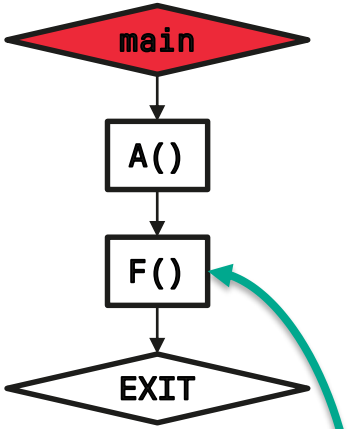


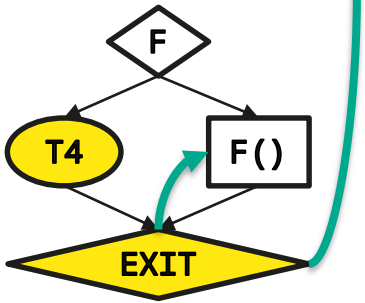


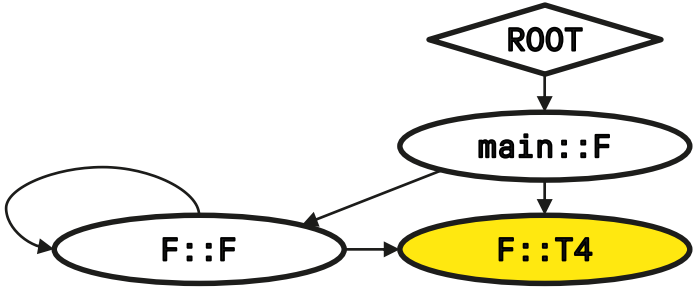


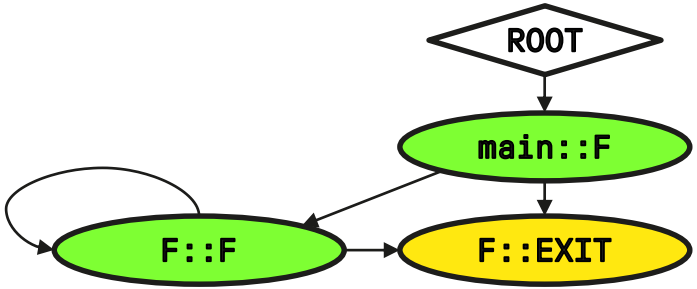


Выход из рекурсии

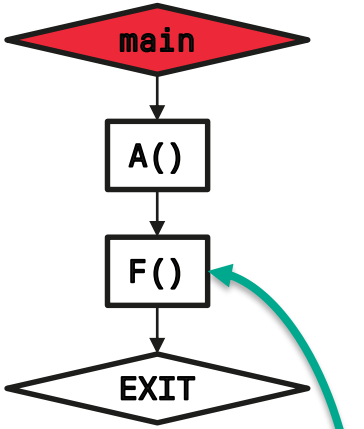


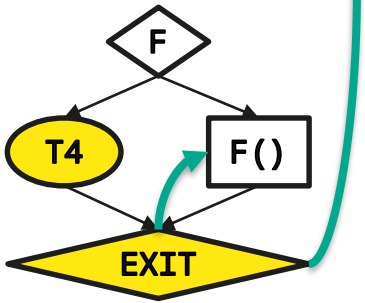


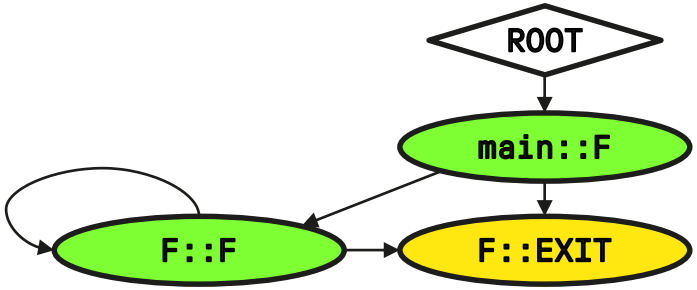


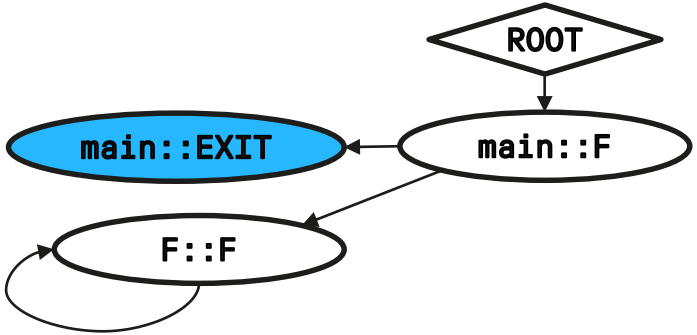


Выход из рекурсии









Результаты работы

Предложена структура данных и алгоритмы над ней, реализующие описанный способ мониторинга потока управления

Предложенная структура данных отражает все возможные стеки вызовов, которые может осуществить поток управления, включая рекурсивные вызовы

Предложенная модель является расширяемой для других случаев нарушения потока управления

.

Спасибо за внимание!

Ссылка на страницу доклада

Данила Пучкин

Разработчик-исследователь
Группа Системных Исследований
Департамент Перспективных Технологий

danila.puchkin@kaspersky.com

kaspersky