Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Ответ защищён с оценкой \_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

к. ф.-м. н.\_\_\_\_\_\_\_\_\_ Крючкова Е.Н.

Отчет

по лабораторным работам №5-6

«Работа с флагом интерпретации»

по дисциплине «Теория языков программирования и методы трансляции»

Студент группы ПИ-81 \_\_\_\_\_\_\_\_\_\_\_ Иванков Д.В.

Преподаватель доцент, к. ф.-м. н.\_\_\_\_\_\_\_ Крючкова Е.Н

Барнаул 2022

**Задание:**

Программа: главная программа языка С++. Допускается описание функций без параметров, функции возвращают значение.

Типы данных: short int ( в том числе short , long, long long) .

Операции: все арифметические, сравнения.

Операторы: присваивания и for простейшей формы.

Операнды: простые переменные, именованные константы и константы.

Константы: целые в 10 c/c и 16 c/c, в том числе длинные.

**Выполнение:**

Перечень СД, в которых используется флаг интерпретации:

1. Описание функций. Заголовки функций интерпретируются, тело main всегда интерпретируется. Нужно сохранить позицию тела функции, чтобы можно было их интерпретировать при вызове функций.
2. Вызов функции. При вызове функции нужно сохранить текущий контекст, установить контекст вызываемой функции, запустить тело функции, восстановить сохранённый контекст.
3. Оператор for. Я расставил метки на места условия, инкремента и тела и перехожу по ним в соответствии с флагом интерпретации.
4. Выражения. При отключённом флаге никакие семантические действия с выражениями не выполняются.
5. Составной оператор. При отключённом флаге составной оператор не добавляется в дерево.
6. Оператор return возвращает значение при включённом флаге интерпретации. После выполнения оператора return флаг сбрасывается и последующий код функции не интерпретируется.

При отключённом флаге интерпретации выполнение всех семантических подпрограмм отключается. Реализовано с помощью добавления следующего блока в начало всех семантических подпрограмм:

if (!flagInterpret)

return;

В некоторых СД используется локальный флаг интерпретации для сохранения глобального и выполнения необходимых действий:

1. Описание функций. Нужно включить при считывании заголовков функций.
2. Оператор for. Используется для соблюдения вложенности, а также для правильного перехода между метками.

Проверим, как работает интерпретатор с реализованным флагом интерпретации на следующем примере:

int function()

{

int k = 300;

return k;

}

int main()

{

int h = function();

for (int i = 0; i < 0; i++)

{

int d = 14;

}

int t = 0;

for (int i = 0; i < 4; i++)

{

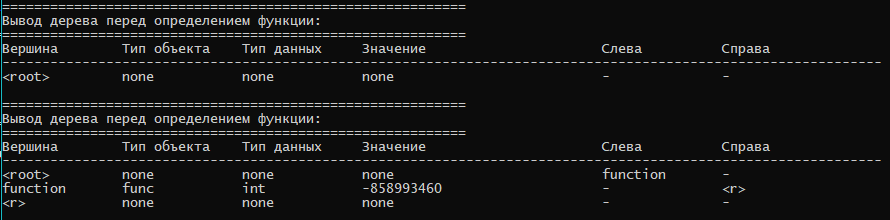
t = t + i;

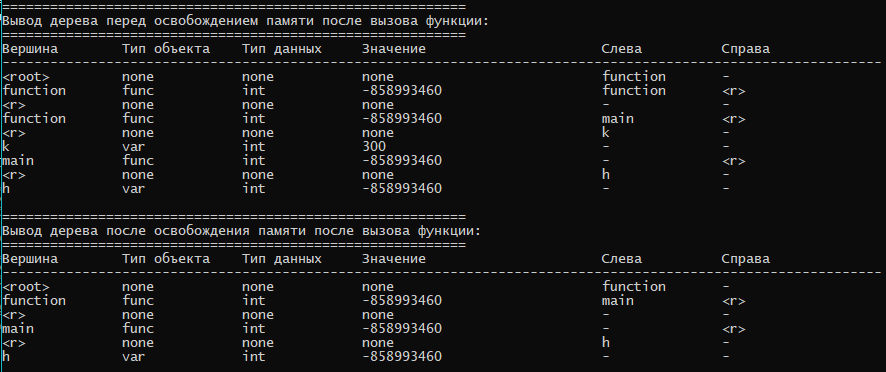
}

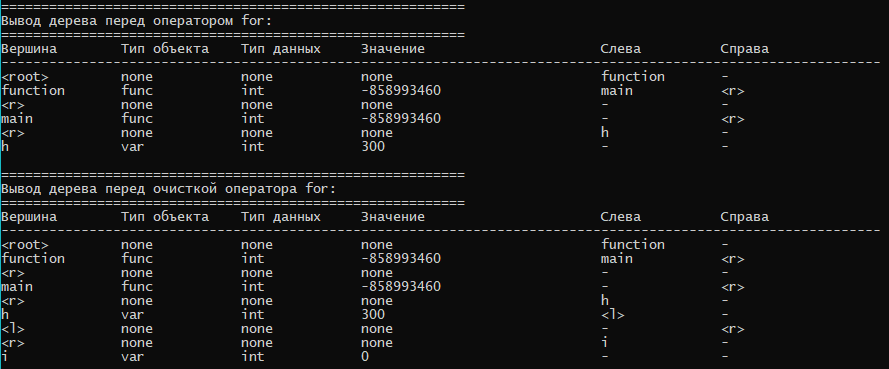
return 1;

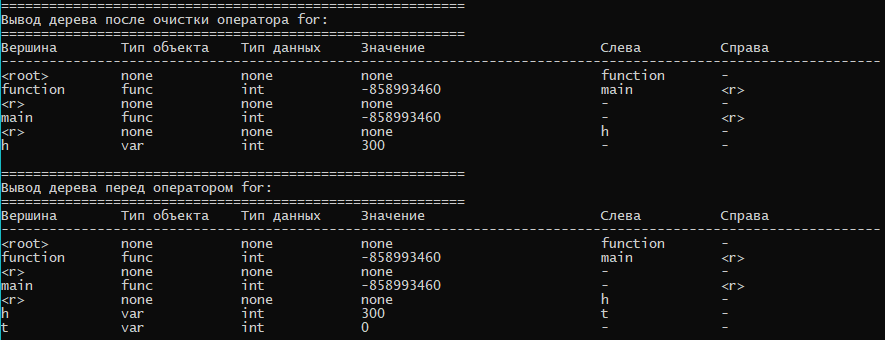
}

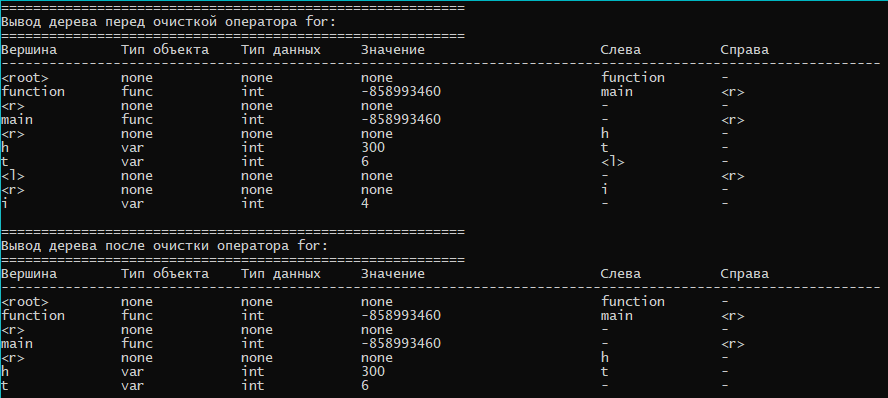
Вывод программы:

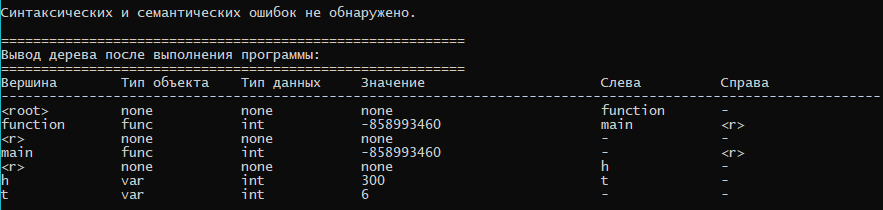












Видно, что первый for не обрабатывался, так как не выполняется условие i < 0 при i = 0;

Также видно, что при описании функции function тело функции не интерпретировалось, а при вызове функции – интерпретируется.

Функция main также интерпретировалась.

Пример для проверки рекурсии и отключения флага интерпретации после оператора return:

int param;

int factor() {

for (int i = 1; param < 2; i++) {

return 1;

}

for (int i = param; i <= param; i++) {

int tmp = param;

param = param - 1;

return tmp \* factor();

param = 1000 \* tmp; // это пример кода который выполняться не должен

}

}

int main() {

param = 5;

int res = factor(); // 5! = 120

param = 4;

res= res + factor(); // проверим результат 5! + 4! = 144

return 0;

param = 100; // не выполняется из-за предшествующего return

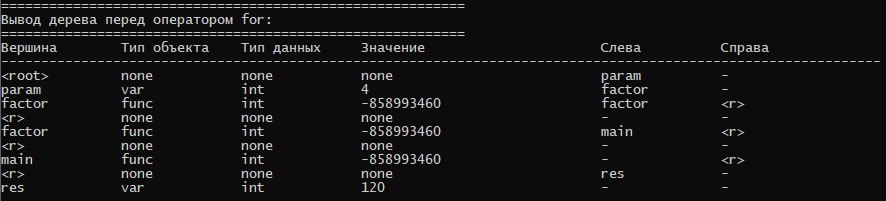
return factor(); // 100! не выполняется

}

Вывод программы:

Программа вывела много ненужно текста из-за того, что в функции используется два цикла и рекурсия, поэтому я прикреплю только значимые фрагменты:

Вывод дерева перед первым оператором for при param = 4. То есть 5! уже посчитался и записался в переменную res. После чего присвоили переменной param значение 4 и снова запустили рекурсивную функцию factor():



Вывод программы после выполнения программы. Посчитался 4!, сложился с 5! и ответ записался в переменную res. Видно, что main вернул значение 0, то есть выполнился оператор return 0; 100! не считался и не возвращался функцией main: