構文木で使用するノードの一覧

C0 言語の処理系は、図１のソースコードを図２の構文木（一部省略）に変換する。

```
int grobal_var = 1; //グローバル変数

int main() {

	int local_var1 = 1; //ローカル変数
	int local_var2 = 2;

	if (grobal_var == local_var) {
		print(”true”);
	} else {
		print(”false”);
	}

	while (local_var1 < local_var2) {
		local_var1++;
	}

	user_func(local_var1);

	return local_var2;
}

int user_func(int parameter) {
	print(”%d”, parameter);
}
```

図１　C0 言語のソースコードのサンプル

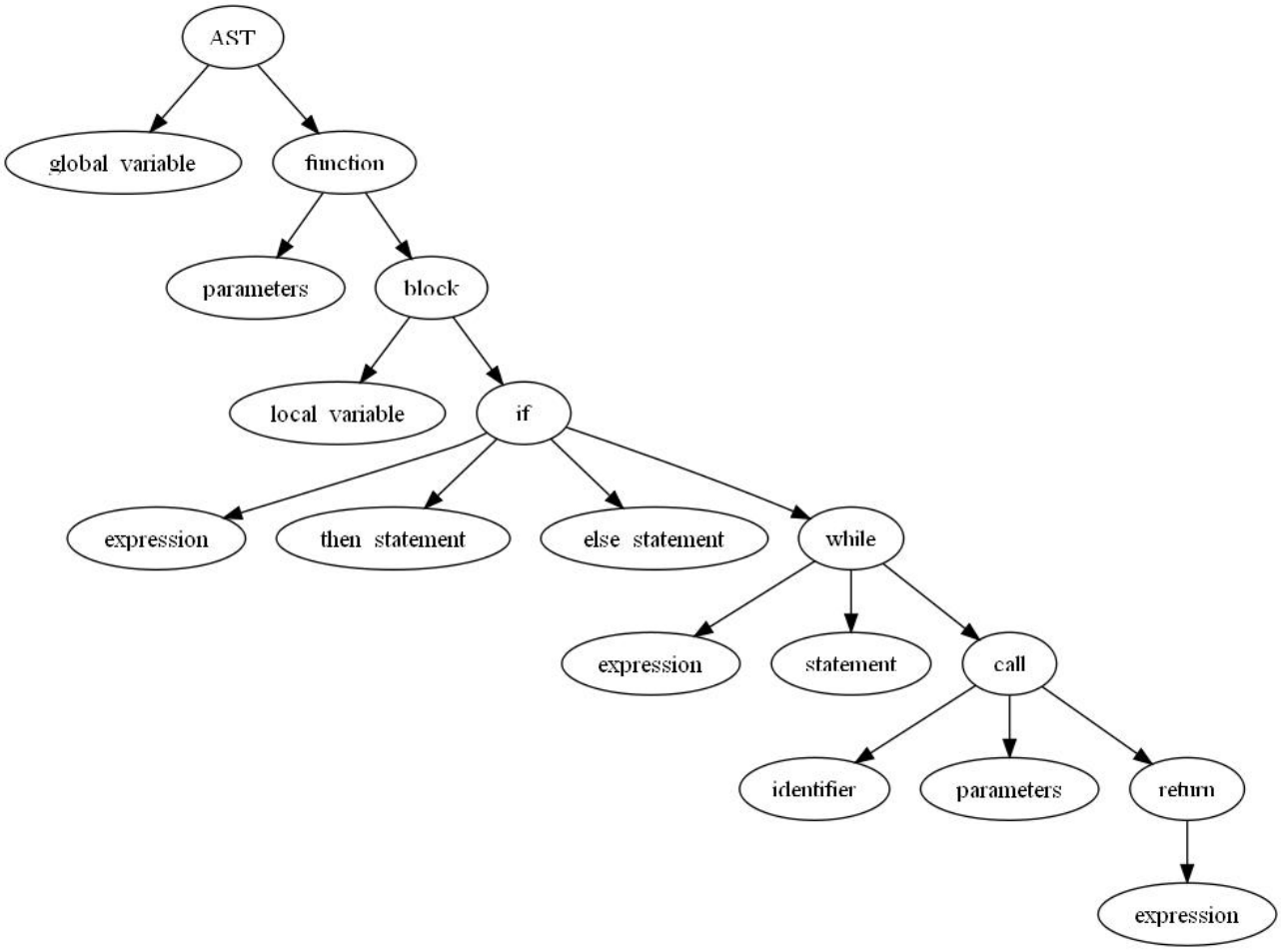
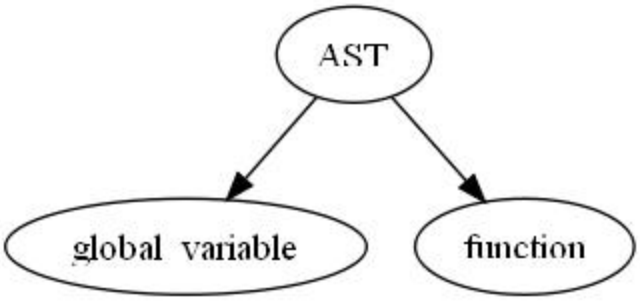


図１　C0 言語処理系の構文木のサンプル

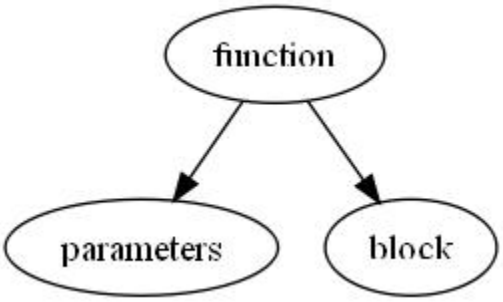
各文の構文木を示す。これらを組み合わせて、ソースコード全体の構文木を作る。

AST



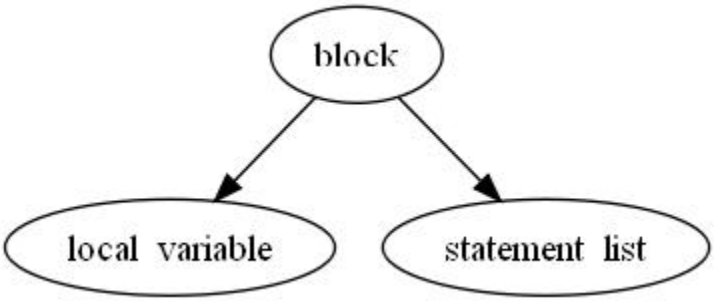
プログラム全体の初期処理。
グローバル変数のリストと main 関数のリンクを持つ。

関数



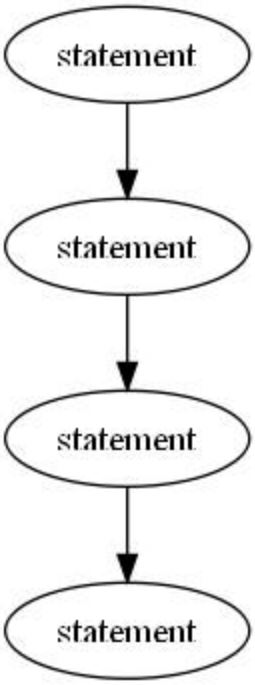
引数のリストとブロック文のリンクを持つ。

複合文



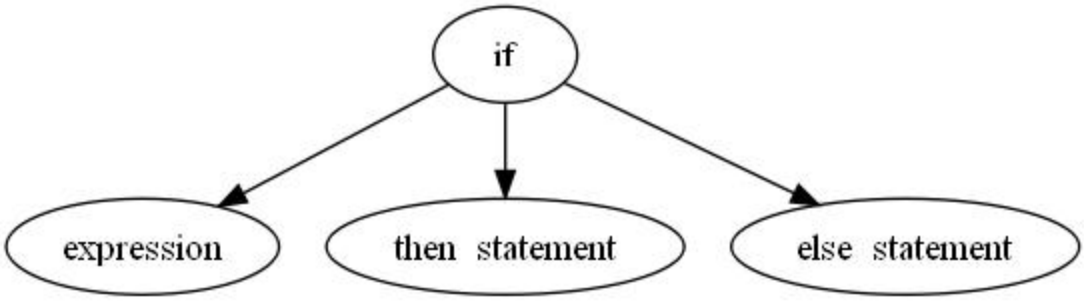
ローカル変数のリストと文のリストのリンクを持つ。

文のリスト



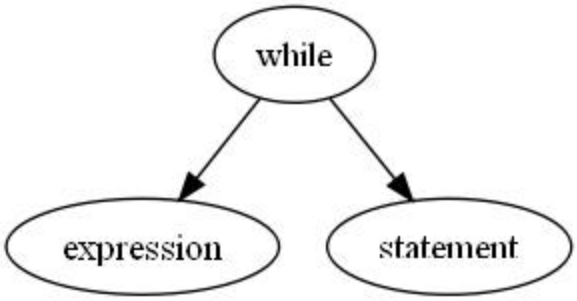
　ブロック文の中では、個々の文をリストの形で保持する。

if 文



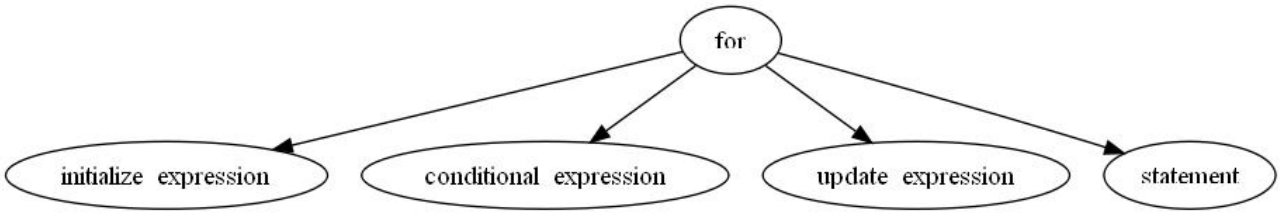
　条件式と then 枝（条件式が真の場合、実行される）と else 枝（条件が偽の場合、実行される）のリンクを持つ。

while 文



条件式と文のリンクを持つ。

for 文

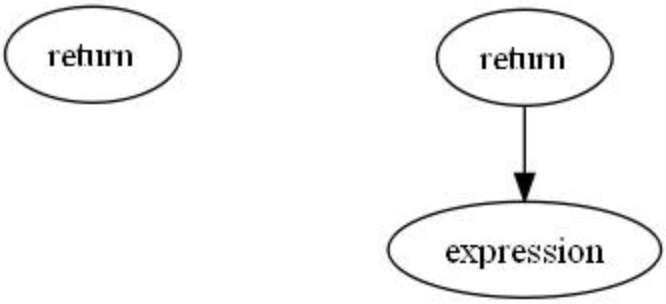


初期化式と制御式と後置き式と文のリンクを持つ。

break 文

break 文のノードはリンクを持たない。

return 文

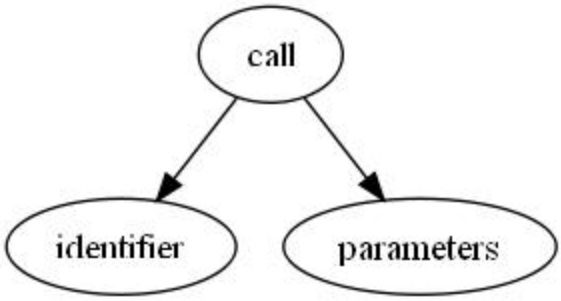


return 文はリンクを持たない場合と式のリンクを持つ場合がある。

空文

　空文のノードはリンクを持たない。

関数呼び出し



　識別子（関数名）と引数のリストのリンクを持つ。