DOCUMENTATION

Constructs:
The following language ***BigF*** does not support function calls at this moment hence the translated C code would always have this format:

```
#include<stdio.h>
int main(){
      (statement);
      (statement);
      (statement);
      ...
}
```

Delimiter:
BigF does uses the same delimiter as the language C, which is ***";"***

Comments:
BigF uses ***"#"*** for single line comments and ***"## (text) ##"*** for multiple line comments

In BigF	In C
##This is a multiline comment It has three lines##	//This is a single line comment /*This is a multiline comment It has three lines*/

Declaration:
f.type.name OR f.type.name = value;
f.type.name[] OR f.type.name[] = {value, value, value, ...};
Type = int, float, char etc.
name = variable name
value = value of variable

In BigF	In C
f.int.a; f.int.b = 10; f.char.c[] = {'a', 'b', 'c'};	Int a; Int b = 10; Char c[] = {'a', 'b', 'c'};

Operations:
assignment: (=), same as C
And operation: (andf), equivalent to &&
Or operation: (orf), equivalent to ||
Not operation: (notf), equivalent to !(operator)

In BigF	In C

`a = 10;`
`b andf c`
`d orf e`
`f != g;`
	`a = 10;`
	`b && c`
	`d || e`
	`f != g;`

Conditional statements:
if (condition) **do** (statement) **fif**
if (condition) **do** (statement) **else** (statement) **fif**
if (condition) **do** (statement) **elif** (statement) **else** (statement) **fif**

In BigF	In C

`if(a == b)`
`do`
`    a+=1;`
`fif`
`----------`
`if(a == b)`
`do`
`    a+=1;`
`else`
`    a-=1;`
`fif`
`----------`
`if(a == b)`
`do`
`    a+=1;`
`elif`
`    a-=1;`
`else`
`    a +=2;`
`fif`
	`if(a == b)`
	`{`
	`    a+=1;`
	`}`
	`----------`
	`if(a == b){`
	`    a+=1;`
	`}`
	`else{`
	`    a-=1;`
	`}`
	`----------`
	`if(a == b){`
	
	`    a+=1;`
	`}`
	`elseif{`
	`    a-=1;`
	`}`
	`else{`
	`    a+=2;`
	`}`

Loop structures:

for loop:
for (f.i = value till required_value; f-) **do** (statement) **ffor**
for (f.i = value till required_value; f+) **do** (statement) **ffor**
default increment/decrement value is 1 but can be changed as: nf-/nf+ where n = 1, 2, 3, ...
f.i in case of previously declared variables, f.int.i in case of variable declared inside loop

while loop:
while (condition) **do** (statement) **fwhile**

dowhile:
yeet (statement) **till** (condition)

In BigF	In C

`for(f.int.i=0 till 10; f+)`
`do`
`    i = i + 2;`
`ffor`
	`for(int i = 0; i < 10; i++)`
	`{`
	`    i = i + 2;`
	`}`

Input / Output:
fin(variable);
fout(variable) OR fout("string");
In BigF, the user doesn't have to specify the data type of the variable, the compiler does it automatically/

In BigF	In C
`fout('Enter number');` `fin(number);`	`printf("Enter number");` `scanf("%d", number):`

Main procedure in BigF:

```
BigF main()
oof:
      (statement);
      (statement);
      (statement);
      ...
endoof:
```

Sample program in BigF

A sample program that checks whether the number provided by user is a perfect square:

In Bigf

```
Bigf main()
oof:
	f.int.number;
	f.int.i;
	f.int.flag;
	flag = 0;

	fout('Enter number to be checked: ');
	fin(number);

	for(f.i = 0 till number, f+)
	do
		if(i*i == number)
		do
			flag = flag + 1;
			break;
		fif
	ffor

	##output##
	if(flag != 0)
	do
		fout(' Value ', number, ' is a perfect square');
	else
		fout(' Value ', number, ' is not a perfect square');
	fif
endoof:
```

Translated in C:

In C

```
#include<stdio.h>
int main() {
	int number;
	int i;
	int flag;
	flag = 0;


	printf("Enter number to be checked: ");
	scanf("%d", &number);

	for(i = 0; i <= number; i++){
		if(i*i == number){
			flag = flag + 1;
			break;
```

```
            }
        }

        //output
        if(flag != 0){
              printf("valude %d is a perfect square", number);
        }else{
              printf("valude %d is not a perfect square", number);
        }
}
```