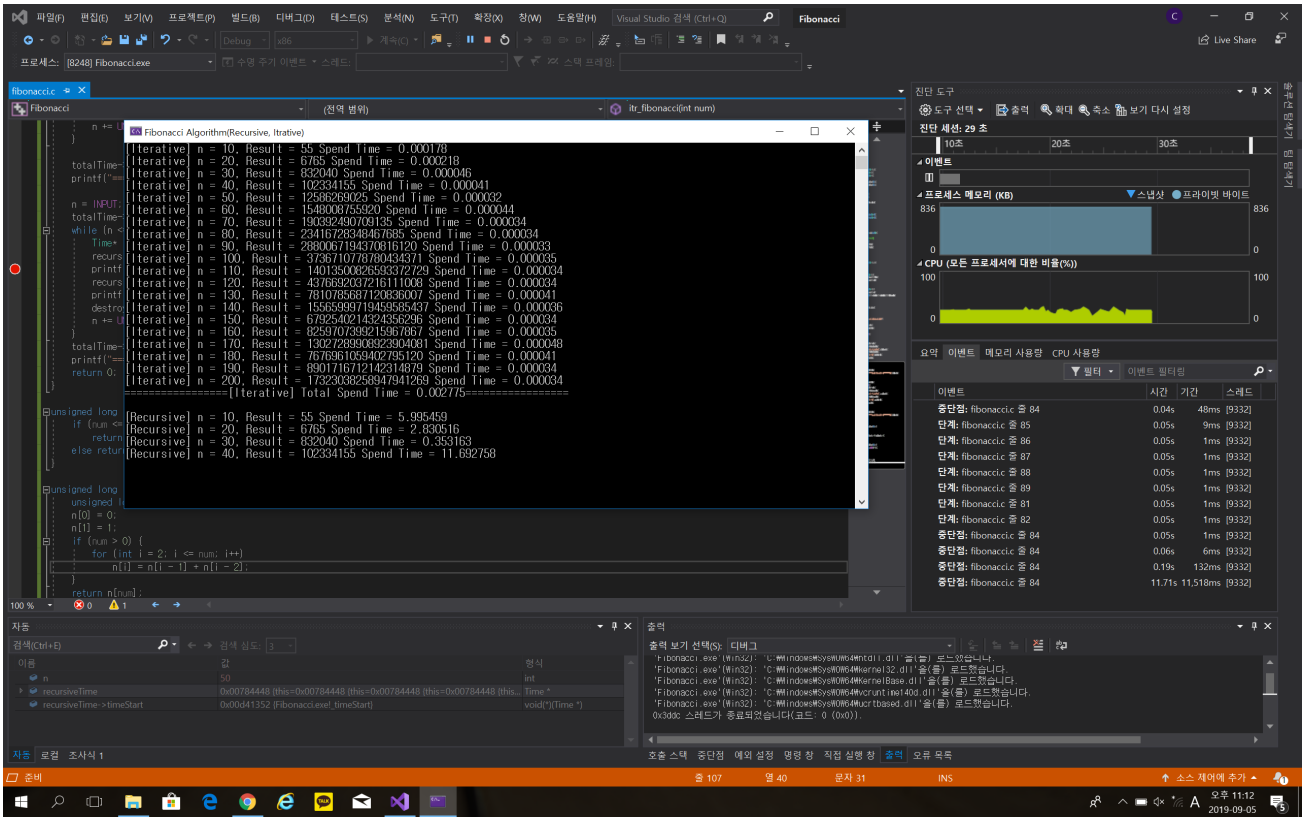
1. Source Code

```
/**
	Soongsil University CSE Algorithm: fibonacci.c
	Author : Kim Byoung June
**/
// Header Declaration
#include <stdio.h>
#include <time.h>
#include <Windows.h>

// Constant Declaration
#define INPUT 10	// Start number
#define OUTPUT 200	// Finish number
#define UNIT 10		// Calculate Range

// Method declaration: Time class
void destroyTime(struct Time* time_ptr);
void setStart(struct Time* this);
void setFinish(struct Time* this);
double getTime(struct Time* this);
void toString(struct Time* this);

// Method declaration: Fibonacci algorithm
long double rec_fibonacci(int num);
long double itr_fibonacci(int num);

// Time object abstract
typedef struct Time {
	struct Time* this;
	LARGE_INTEGER timefreq, start, end;
	double time;
	void (*setStart)(struct Time* this);
	void (*setFinish)(struct Time* this);
	double (*getTime)(struct Time* this);
	void (*toString)(struct Time* this);
}Time;

// Time object constructor
Time* newTime() {
	Time* temp = (Time*)malloc(sizeof(Time));
	temp->this = temp;
	temp->setStart = setStart;
	temp->setFinish = setFinish;
	temp->getTime = getTime;
	temp->toString = toString;
	return temp;
}

// Time object destructor
void destroyTime(struct Time* time_ptr) { free(time_ptr); }

// Time object method: Start time setter
void setStart(struct Time* this) {
	QueryPerformanceFrequency(&this->timefreq);
	QueryPerformanceCounter(&this->start);
}

// Time object method: Finish time setter
void setFinish(struct Time* this) {
```

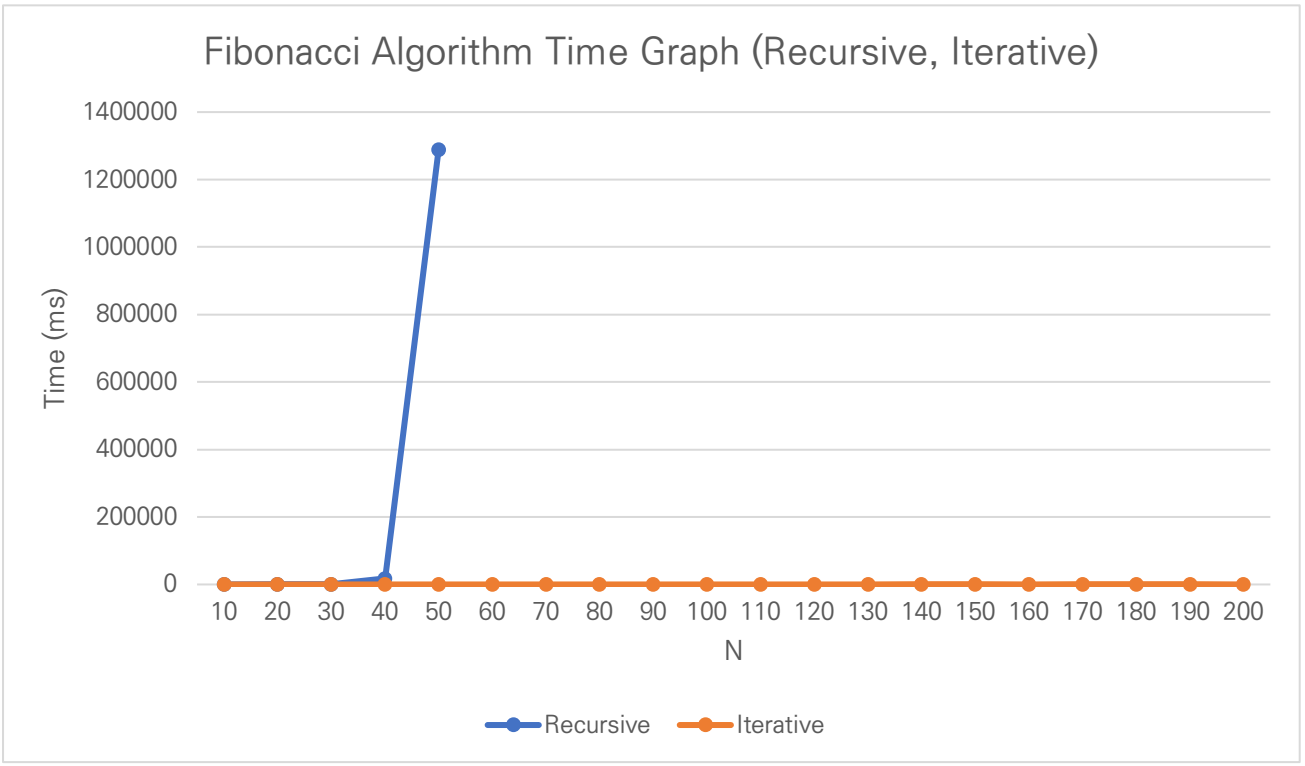
```
	QueryPerformanceCounter(&this->end);
	// 1s = 1000ms
	this->time = (double)(this->end.QuadPart - this->start.QuadPart) * 1000 / this-
>timefreq.QuadPart;
}

// Time object method: Time getter
double getTime(struct Time* this) { return this->time; }

// Time object method: Print time
void toString(struct Time* this) {
	printf("Spend Time = %.4lfms\n", this->time);
}

// Main method
int main(void) {
	// Set Title
	system("title Fibonacci Algorithm(Recursive, Iterative): 20162448 컴퓨터학부 가반
김병준");
	Time* totalTime = newTime();
	Time* algorithmTime = newTime();

	// Iterative fibonacci algorithm measurement
	totalTime->setStart(totalTime);
	for(int n = INPUT; n <= 200; n += UNIT) {
		algorithmTime->setStart(algorithmTime);
		printf("[Iterative] n = %3d, Result = %42.0lf, ", n, itr_fibonacci(n));
		algorithmTime->setFinish(algorithmTime);
		algorithmTime->toString(algorithmTime);
	}
	totalTime->setFinish(totalTime);
	printf("[Iterative]
============================================================== Total Time
= %.4lfms\n\n", totalTime->getTime(totalTime));

	// Recursive fibonacci algoritm measurement
	totalTime->setStart(totalTime);
	for (int n = INPUT; n <= 200; n += UNIT) {
		algorithmTime->setStart(algorithmTime);
		printf("[Recursive] n = %3d, Result = %42.0lf, ", n, rec_fibonacci(n));
		algorithmTime->setFinish(algorithmTime);
		algorithmTime->toString(algorithmTime);
	}
	totalTime->setFinish(totalTime);
	printf("[Recursive]
============================================================== Total Time
= %.4lfms\n\n", totalTime->getTime(totalTime));
	destroyTime(algorithmTime);
	return 0;
}

/**
	Fibonacci Algorithm
		f(0) = 0, f(1) = 1
		f(n) = f(n - 1) + f(n - 2)
	Precondition:  Input is integer
	Postcondition: Output is Integer which casting type is long double
**/
```

```
// Fibonacci algorithm method: Recursive
long double rec_fibonacci(int num) {
	if (num <= 1) return num;
	else return rec_fibonacci(num - 1) + rec_fibonacci(num - 2);
}

// Fibonacci algorithm method: Iterative
long double itr_fibonacci(int num) {
	long double n[OUTPUT + 1] = { 0, 1, 0 }; // f(0) ~ f(200)
	if (num > 1) {
		for (int i = 2; i <= num; i++)
			n[i] = n[i - 1] + n[i - 2];
	}
	return n[num];
}
```

2. Debug Screenshot



3. Time Graph



- Recursive의 경우 N 〉= 50부터 시간을 20분을 초과함으로 측정하지 않음.