운영체제 프로젝트 보고서

스도쿠 검사

Yu, ho Geun (유 호 근)

융합 소프트웨어 학과

한림대학교
Hallym University

2017년도

목 차

요 약 문

1. 스도쿠 검사하기 위한 프로그램

스도쿠의 각 가로 열, 세로 열 9가지의 3x3 의 크기의 박스들을 검사하는 프로그램이다.

2. 연구 목적 및 필요성

스도쿠의 검사하는 프로그램을 만드는 이유는 스도쿠를 풀었을 때 1서부터 9까지의 숫자가 겹쳐지는지 검사하기 위해

3. 연구 내용 및 결과

스도쿠를 직접 입력을 해서 각 가로, 세로열과 9가지 3x3 의 크기 박스열의 1부터9 까지 숫자가 겹치는지 검사하고 결과 출력해준다.

6. 기대효과

스도쿠의 직접 풀어보고 잘못 기입한곳이 있는지 검사를 할수 있다.

제 1 장 서 론

제 1 절 스도쿠의 정의

스도쿠는 마방진의 일종으로 9x9 칸에서 진행 되는 숫자 퍼즐 게임이다 18세기 스위스 수학자 레온하르트 오일러가 창안한 라틴 방진 에 기초해 미국의 걱축가 하워드 간즈가 넘버 플레이스 라는 이름으로 소개했다. 이후 일본에 니코리사의 잡지에 스도쿠라는 이름을 붙여 수록하면서 대중화 되었다

스도쿠의 규칙으로는 구성하는 칸이 총 81 칸 3x3칸이 9개로 세분화 되어 있다. 그런데 여기서 지켜야 할 룰이 있다. 각 가로 세로줄이 1부터 9까지 의 수가 중복 없이 들어 가야 하고 3x3 칸 안에 1부터 9까지의 수가 중복 해서 들어 가지 말아야 한다.

보통 주어지는 문제에서는 일부분만 숫자가 체워져 있기 때문에 위의 규칙을 바탕으로 나머지 칸들의 숫자를 유추해야 한다. 기차나 비행기에서 시간 때우기로 딱 좋은 퍼즐이다. 신문 자투리로 나오기도 하며 비슷한 퍼즐들과 함께 잔뜩 모아놓은 소형 책제도 심심치 않게 볼수 있다. 한국에서의 인지도도 낮지는 않다

ex) 스도쿠의 모양

			2		7			
	9						1	
					9			
6		4						5
				2				
9						2		1
			8					
	1						5	
			7		5			

제 2 장 본 론

제1절 프로그램 소개

1. 프로그램 용도

신문지나 잡지 마지막 장에 있는 두뇌 게임인 스도쿠를 하게 되는데 스도쿠의 가로 열과 세로 열과 3x3 박스 안에 있는 숫자들이 1서부터 9까지의 숫자들이 겹치지 않고 존재해야 한다.

그래서 이 프로그램에서는 맨 위에 있는 가로줄 서 9개의 숫자를 쓰면서 9개의 가로줄을 채워 줌으로서 81 개(9x9)의 숫자를 채워 넣게 된다. 그리고 그 넣은 숫자들을 가로 별로 검사를 해서 몇 번째 가로줄에서 어떤 좌표 값에 숫자가 겹치는지 출력을 해주고 다음으로 세로줄의 숫자를 검사를 해서 몇 번째 세로줄에서 어떤 좌표 값의 숫자가 겹치게 되는지 출력을 해주고 각 3x3(9개)의 9 가지 박스 안에 있는 숫자들을 검사를 해서 어느 박스의 어떤 좌표 값의 겹쳐지는 숫자가 겹치게 되는지 출력을 해주는 프로그램이다.

제2절 프로그램 소개

2. 소스코드 설명

```
void rows(int val[][9]);
void cols(int val[][9]);
void box1(int val[][9]);
void box2(int val[][9]);
void box3(int val[][9]);
void box4(int val[][9]);
void box5(int val[][9]);
void box6(int val[][9]);
void box7(int val[][9]);
void box8(int val[][9]);
void box9(int val[][9]);
```

-함수 선언

```
for(row = 0; row < 9; row++){
        printf("%d, row >>",row);
        for(col = 0; col < 9; col++)
                scanf("%d",&val[row][col]);
}
printf("------print------\n");
for(row = 0; row < 9; row++){
        for(col = 0; col < 9; col++){
                printf("%d ",val[row][col]);
        }
        printf("\n");
}
printf("-----------------\n");
```

-스도쿠 9x9 배열 입력 및 출력

val의 int형 2차원 배열에 9x9의 크기로 입력을 받는데 한 row 값에 9개의 int형 값을 scanf를 이용하여 키보드의 입력값을 받아 저장을 해준다.

다음 2중 포문에서는 위에서 저장한 int형 2차원 배열 var을 9x9 크기의 배열을 row별로 출력을 해준다.

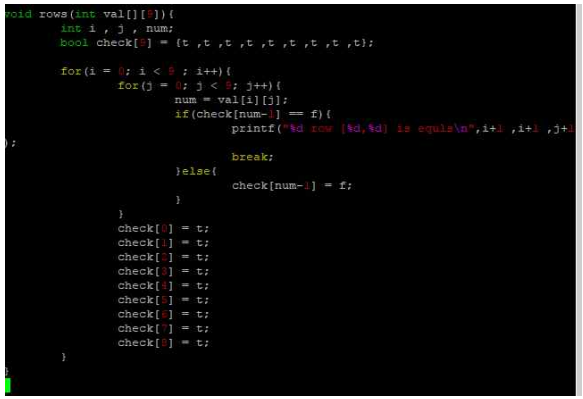
```
pid_t pidval[size], pid;
int run ,child_pid,status;

for(run = 0;run < size; run++){
        pidval[run] = fork();

        if(pidval[run] < 0)
                return 0;
        else if(pidval[run] == 0){
                pid = fork();
                if(pid == 0){
                        switch(run){
                        case 1:
                        printf("------row------\n");
                        rows(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 2:
                        printf("------col------\n");
                        cols(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 3:
                        printf("------box1------\n");
                        box1(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 4:
                        printf("------box2------\n");
                        box2(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 5:
                        printf("------box3------\n");
                        box3(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 6:
                        printf("------box4------\n");
                        box4(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 7:
                        printf("------box5------\n");
                        box5(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;
```

```
                        case 7:
                        printf("------box5------\n");
                        box5(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 8:
                        printf("------box6------\n");
                        box6(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 9:
                        printf("------box7------\n");
                        box7(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 10:
                        printf("------box8------\n");
                        box8(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;

                        case 11:
                        printf("------box9------\n");
                        box9(val);
                        child_pid=wait(&status);
                        printf("--------------\n");
                        break;
                        }
                }
                child_pid=wait(&status);
                exit(0);
        }
}
```

- 포크를 이용하여 함수 실행

  포크를 생성을 하고 각 포크의 값에 따른 선언한 함수들을 실행을 해준다.

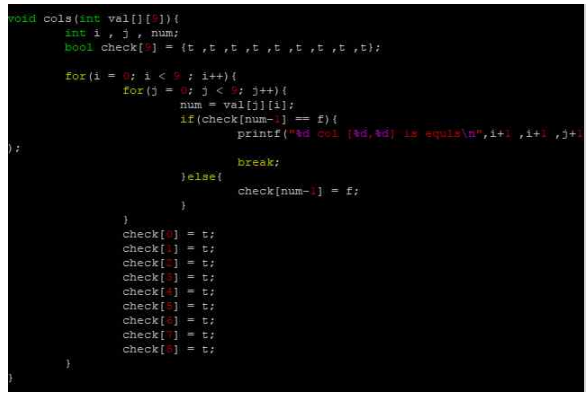
  여기서 wait를 이용을 하여 자녀 프로세스가 실행되는 동안 다른 프로세스 자녀 프로세스가 작동하는 것을 방지해 준다.



```
void rows(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 0; i < 9 ; i++){
                for(j = 0; j < 9; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("%d row [%d,%d] is equls\n",i+1 ,i+1 ,j+1
);
                                break;
                        }else{
                                check[num-1] = f;
                        }
                }
                check[0] = t;
                check[1] = t;
                check[2] = t;
                check[3] = t;
                check[4] = t;
                check[5] = t;
                check[6] = t;
                check[7] = t;
                check[8] = t;
        }
}
```

-row값에 대한 검사

row검사하는 함수를 구현한 부분이다. 여기서 bool 형을 typedef 로 지정 f는 false t는 true로 의미하는 1차원 배열에 9개의 true 값을 넣어 두고 2중 포문을 사용하면서 9x9 크기의 배열을 검사를 하는데 check 이 변수가 의미 하는 것은 각 배열이 보유하고 있는 숫자를 의미 하고 1개가 찾아지면 false 로 바뀌고 또 같은 값이 나오게 되면 if(check[num-1] == f) 이 if 문이 실행을 하게 되면서 현재 몇 번째 row 인지와 좌표 값을 가르켜 주는 함수이다.



```
void cols(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 0; i < 9 ; i++){
                for(j = 0; j < 9; j++){
                        num = val[j][i];
                        if(check[num-1] == f){
                                printf("%d col [%d,%d] is equls\n",i+1 ,i+1 ,j+1
);
                                break;
                        }else{
                                check[num-1] = f;
                        }
                }
                check[0] = t;
                check[1] = t;
                check[2] = t;
                check[3] = t;
                check[4] = t;
                check[5] = t;
                check[6] = t;
                check[7] = t;
                check[8] = t;
        }
}
```

-col값에 대한 검사

col검사하는 함수와 같이 구현한 부분이다. 여기서 bool 형을 typedef 로 지정 f는 false t는 true로 의미하는 1차원 배열에 9개의 true 값을 넣어 두고 2중 포문을 사용하면서 9x9 크기의 배열을 검사를 하는데 check 이 변수가 의미 하는 것은 각 배열이 보유하고 있는 숫자를 의미 하고 1개가 찾아지면 false 로 바뀌고 또 같은 값이 나오게 되면 if(check[num-1] == f) 이 if 문이 실행을 하게 되면서 현재 몇 번째 col 인지와 좌표 값을 가르켜 주는 함수이다.

```
void box1(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 0; i < 3 ; i++){
                for(j = 0; j < 3; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("1st box [%d,%d] is equls\n" ,i+1 ,j+1);
                                continue;
                        }else{
                                check[num-1] = f;
                        }
                }
        }
}

void box2(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 0; i < 3 ; i++){
                for(j = 3; j < 6; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("2nd box [%d,%d] is equls\n",i+1 ,j+1);
                                continue;
                        }else{
                                check[num-1] = f;
                        }
                }
        }
}

void box3(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 0; i < 3 ; i++){
                for(j = 6; j < 9; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("3rd box [%d,%d] is equls\n",i+1 ,j+1);
                                continue;
                        }else{
                                check[num-1] = f;
                        }
                }
        }
}
```

-첫 번째 줄에 대한 박스 검사

이 함수들은 맨 위에 있는 3x3 배열 3개를 검사하는 것을 구현한 것들이다.

박스 함수에서는 위에 row 와 col을 검사하는 함수와 같이 하지만 이 함수에서는 전체의 배열을 검사하는거 것이 아니라 0에서 3 까지 3에서 6까지 6에서 9까지의 배열을 3x3의 크기로 잘라서 검사를 하는 것이다. 위 와 같이 1에서 9까지 숫자가 존재한다면 f(false)로 바꿔주고 또 같은 숫자가 만나게 되면 if 의 조건이 되기 때문에 몇 번째 박스의 어디 좌표가 잘못이 됐는지 출력을 해준다.

```
void box4(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 0; i < 6 ; i++){
                for(j = 0; j < 3; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("4th box [%d,%d] is equls\n",i+1 ,j+1);
                                continue;
                        }else{
                                check[num-1] = f;
                        }
                }
        }
}

void box5(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 3; i < 6 ; i++){
                for(j = 3; j < 6; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("5th box [%d,%d] is equls\n",i+1 ,j+1);
                                continue;
                        }else{
                                check[num-1] = f;
                        }
                }
        }
}

void box6(int val[][9]){
        int i , j , num;
        bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

        for(i = 3; i < 6 ; i++){
                for(j = 6; j < 9; j++){
                        num = val[i][j];
                        if(check[num-1] == f){
                                printf("6th box [%d,%d] is equls\n",i+1 ,j+1);
                                continue;
                        }else{
                                check[num-1] = f;
                        }
                }
        }
}
```

-두 번째 줄에 대한 박스 검사

이 함수들은 첫 번째 줄에 대한 박스 검사와 같이 가운데 줄에 있는 3x3 배열 3개를 검사하는 것을 구현한 것들이다.

박스 함수에서는 위에 row 와 col을 검사하는 함수와 같이 하지만 이 함수에서는 전체의 배열을 검사하는거 것이 아니라 0에서 3 까지 3에서 6까지 6에서 9까지의 배열을 3x3의 크기로 잘라서 검사를 하는 것이다. 위 와 같이 1에서 9까지 숫자가 존재한다면 f(false)로 바꿔주고 또 같은 숫자가 만나게 되면 if 의 조건이 되기 때문에 몇 번째 박스의 어디 좌표가 잘못이 됐는지 출력을 해준다.

```
void box7(int val[][9]){
	int i , j , num;
	bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

	for(i = 6; i < 9 ; i++){
		for(j = 0; j < 3; j++){
			num = val[i][j];
			if(check[num-1] == f){
				printf("7th box [%d,%d] is equls\n",i+1 ,j+1);
				continue;
			}else{
				check[num-1] = f;
			}
		}
	}
}

void box8(int val[][9]){
	int i , j , num;
	bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

	for(i = 6; i < 9 ; i++){
		for(j = 3; j < 6; j++){
			num = val[i][j];
			if(check[num-1] == f){
				printf("8th box [%d,%d] is equls\n",i+1 ,j+1);
				continue;
			}else{
				check[num-1] = f;
			}
		}
	}
}

void box9(int val[][9]){
	int i , j , num;
	bool check[9] = {t ,t ,t ,t ,t ,t ,t ,t ,t};

	for(i = 6; i < 9 ; i++){
		for(j = 6; j < 9; j++){
			num = val[i][j];
			if(check[num-1] == f){
				printf("9th box [%d,%d] is equls\n",i+1 ,j+1);
				continue;
			}else{
				check[num-1] = f;
			}
		}
	}
}
```

-세 번째 줄에 대한 박스 검사

이 함수들은 첫 번째 줄에 대한 박스 검사와 두 번째 줄에 대한 박스 검사와 같이 마지막 줄에 있는 3x3 배열 3개를 검사하는 것을 구현한 것들이다.

박스 함수에서는 위에 row 와 col을 검사하는 함수와 같이 하지만 이 함수에서는 전체의 배열을 검사하는거 것이 아니라 0에서 3 까지 3에서 6까지 6에서 9까지의 배열을 3x3의 크기로 잘라서 검사를 하는 것이다. 위 와 같이 1에서 9까지 숫자가 존재한다면 f(false)로 바꿔주고 또 같은 숫자가 만나게 되면 if 의 조건이 되기 때문에 몇 번째 박스의 어디 좌표가 잘못이 됐는지 출력을 해준다.

3. 결과

```
20135332@ciclab:~/OS$ ./sudoku
0, row >>1 2 3 4 5 6 7 8 9
1, row >>9 8 7 2 3 1 4 5 6
2, row >>2 3 4 5 6 7 8 9 1
3, row >>1 9 2 8 3 7 4 6 5
4, row >>1 1 1 1 1 1 1 1 1
5, row >>8 7 6 5 3 9 2 1 4
6, row >>2 2 2 4 4 4 8 8 8
7, row >>2 2 2 4 4 4 8 8 8
8, row >>2 2 2 4 4 4 8 8 8
-----print-----
1 2 3 4 5 6 7 8 9
9 8 7 2 3 1 4 5 6
2 3 4 5 6 7 8 9 1
1 9 2 8 3 7 4 6 5
1 1 1 1 1 1 1 1 1
8 7 6 5 3 9 2 1 4
2 2 2 4 4 4 8 8 8
2 2 2 4 4 4 8 8 8
2 2 2 4 4 4 8 8 8
---------------
```

-입출력 결과

위에 있는 배열은 몇 번째 열에 9개의 숫자를 1~9 까지의 숫자를 입력을 하고 print 라고 출력 되 있는 부분 아래 에 배열은 입력한 값에 대한 것에 출력 값이다.

```
-----row-----
6 row [5,2] is equls
7 row [7,2] is equls
8 row [8,2] is equls
9 row [9,2] is equls
------------ End -----
```

-row 검사 결과

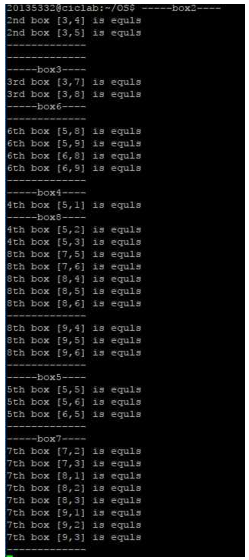
row 검사에서는 row 값에 대한 검사의 결과 값을 몇 번째 row 인지와 틀린 곳의 좌표만 가르켜 주도록 출력이 된다

```
-----col-----
1 col [1,4] is equls
2 col [2,7] is equls
3 col [3,7] is equls
4 col [4,6] is equls
5 col [5,4] is equls
6 col [6,4] is equls
7 col [7,4] is equls
8 col [8,6] is equls
9 col [9,5] is equls
```

-col 검사 결과

위의 row 의 결과값과 같이 col 값에 대한 검사의 결과 값을 몇 번째 col 인지와 틀린 곳의 좌표만 가르쳐 주도록 출력이 된다.

```
-----box1----
1st box [3,1] is equls
1st box [3,2] is equls
------------
```



-박스 검사 결과

박스의 검사에서는 각각의 박스의 오류를 검출을 해주는데 몇 번째 박스에서 몇 번째 좌표에서 값이 같은지 출력을 해주는 것이다.