게임 프로그래밍

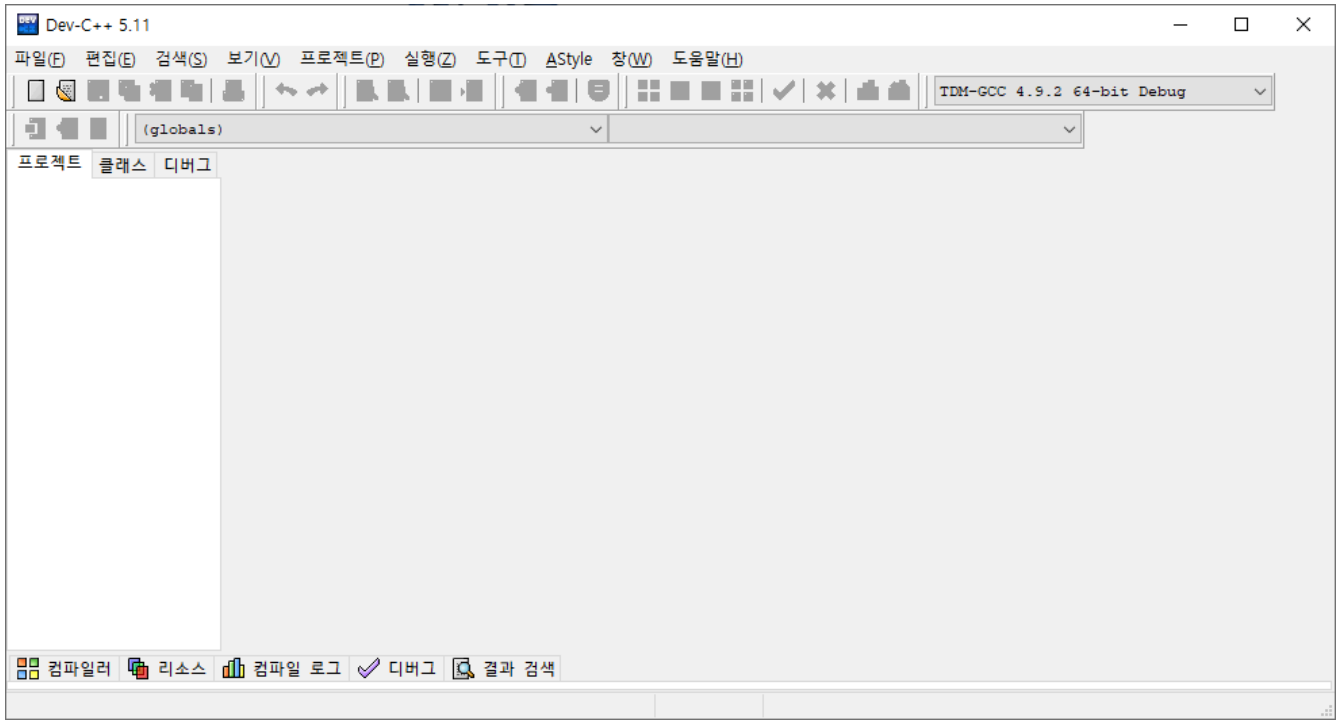
c

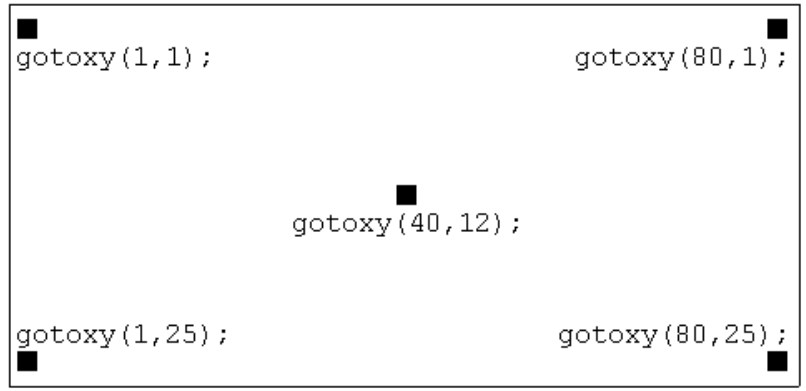
Game Programming

게임기본모듈

KYUNGSUNG UNIVERSITY SINCE 1955

개발 환경 https://sourceforge.net/projects/orwelldevcpp/





커서의 위치 제어

커서의 위치 이동 : 함수 gotoxy를 이용

구분	Visual C++	Turbo C/C++

커서의 위치제어 함수	없음	#include <conio.h>
		gotoxy(int x, int y);

```
void gotoxy(int x, int y)
{
   COORD Pos = {x - 1, y - 1};
   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
```

커서의 위치 제어

```
#include <stdio.h>
#include <windows.h>
void gotoxy(int x, int y);
int main(void)
{
  gotoxy(2,4);
  printf("Hello");
  gotoxy(40, 20);
  printf("Hello");
  return 0;
}

void gotoxy(int x, int y)
{
   COORD Pos = {x - 1, y - 1};
   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
//커서 위치 제어
```

```
#include <stdio.h>
#include <windows.h>
void gotoxy(int x, int y);
int main(void)
{
  for(int i=1;i<=9;i++)
  {
     gotoxy(35, 5+i);
     printf("%d*%d=%2d",3,i,3*i);
  }
  printf("₩n");
  return 0;
}
//3단 출력
```

화면 지우기

구분	Visual C++	Turbo C/C++

커서의 위치제어 함수	#include <stdlib.h>
	system("cls");
		#include <conio.h>
		clrscr();

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  char ch;
  printf("문자를 입력하고 Enter>");
  scanf("%c", &ch);
  system("cls");
  printf("입력된 문자 %c₩n", ch);
  return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main(void)
{
 int i, j;
 for(j=1;j<=9;j++)
 {
  system("cls");
  for(i=1;i<=9;i++)
   printf("%d*%d=%d₩n", j, i, j*i);
  printf("아무키나 누르시오.₩n");
  getch();
  }
 return 0;
}
```

getchar, getche, getch
C언어에서 입력 버퍼 비우기

ASCII code & scan code

- 아스키 코드 : 컴퓨터 내부에서 문자를 처리(또는 전송)하기 위한 일종의 규칙으로, 'a' 라는 문자에 대해서 미리 약속한 코드 값을 의미
- 스캔 코드 : 각각의 키(key)에 대한 코드 값을 의미

- 일반적으로 스캔 코드는 확장키 코드를 말함.
- 2바이트로써 상위 바이트는 스캔 코드이고 하위 바이트는 아스키 코드로 구성.
- 확장키 코드란 1 byte에 해당하는 256개의 아스키코드로 나타낼 수 없는 키를 말하며 화살표 키를 포함하여 Home, End, Page Up, Page Down 등이 있다.

코드	상위 1 byte	하위 1 byte
아스키코드	0	스캔코드
확장 코드	스캔코드	0

스캔 코드와 아스키 코드의 구별 없이 모두 저장하려면 2 byte가 필요

확장키 코드가 입력되었을 경우에는 0 또는 224(0xe0)를 반환하므로 getch를 한 번 더 호출하면 확장 키 코드를 얻을 수 있다.

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
   int chr;
   do
   {
      chr=getch();
      if (chr==0 || chr == 0xe0)
      {
          chr=getch();
          printf("확장키 code=%d₩n", chr);
       }
       else
          printf("아스키 code=%d₩n", chr);
      }while(1);
   return 0;
}
```

```
void move_arrow_key(char key, int *x1, int *y1, int x_b, int y_b)
{
        switch(key)
        {
        case 72:  //위쪽(상) 방향의 화살표 키 입력
                  *y1=*y1-1;
                  if (*y1<1) *y1=1; //y좌표의 최소값
                  break;
        case 75:  //왼쪽(좌) 방향의 화살표 키 입력
                  *x1=*x1-1;
                  if (*x1<1) *x1=1; //x좌표의 최소값
                  break;
        case 77:  //오른쪽(우) 방향의 화살표 키 입력
                  *x1=*x1+1;
                  if (*x1>x_b)  *x1=x_b; //x좌표의 최대값
                  break;
        case 80:  //아래쪽(하) 방향의 화살표 키 입력
                  *y1=*y1+1;
                  if (*y1>y_b)  *y1=y_b; //y좌표의 최대값
                  break;
        default:
                  return;
        }
}
```

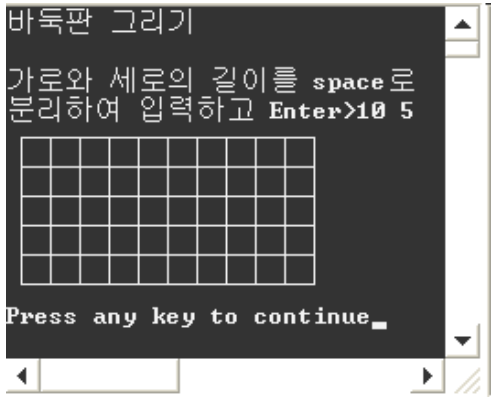
화살표 키	스캔 코드(10진수)	x의 변화	y의 변화
상 (↑)	72	없음	y--;
하 (↓)	80	없음	y++;
좌 (←)	75	x--;	없음
우 (→)	77	x++;	없음

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#define X_MAX 79  //가로(열)방향의 최대값
#define Y_MAX 24  //세로(행)방향의 최대값
void move_arrow_key(char chr, int *x, int *y, int x_b, int y_b);
void gotoxy(int x, int y);
int main(void)
{
        char key;
        int x=10, y=5;
        do
        {
                gotoxy(x, y);
                printf("A");
                key=getch();
                move_arrow_key(key, &x, &y, X_MAX, Y_MAX);
        }while(key!=27);
        return 0;
}
```

사각형 그리기

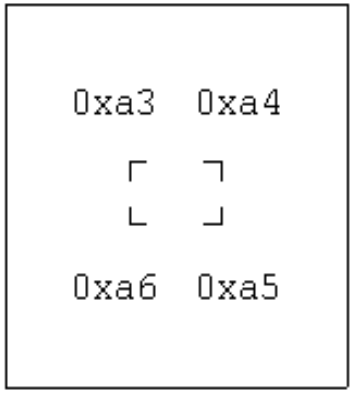
컴퓨터 그래픽에서 화면에 사각형을 표현하는 방법은 꼭짓점에 해당하는 좌표와 이 점들을 연결하는 명령(선)으로 나타내지만 텍스트 모드에서는 이와 같은 방법을 이용하기 어려우므로 확장 완성형 코드(—, | , ┌, ┘ 등)를 연속적으로 사용하여 사각형을 표시하는 방법에 대해서 설명.

우선 정사각형을 표현하는 방법을 설명하고, 이어서 직사각형과 바둑판과 같은 격자 모양을 표현하는 방법을 설명.



정사각형의 표현

화면에 정사각형 모양을 표시하기 위해서 기본적으로 4개의 모서리를 나타내는 기호를 연속적으로 출력하는 방법을 이용



확장 완성형 코드 (10진수)	기호
0xa3(163)	┌
0xa4(164)	┐
0xa5(165)	┘
0xa6(166)	└

확장 완성형 코드를 이용하는 방법

```
printf("%c%c", 0xa6, 0xa3);
printf("%c%c", 0xa6, 0xa4);
printf("\n");
printf("%c%c", 0xa6, 0xa6);
printf("%c%c", 0xa6, 0xa5);
```

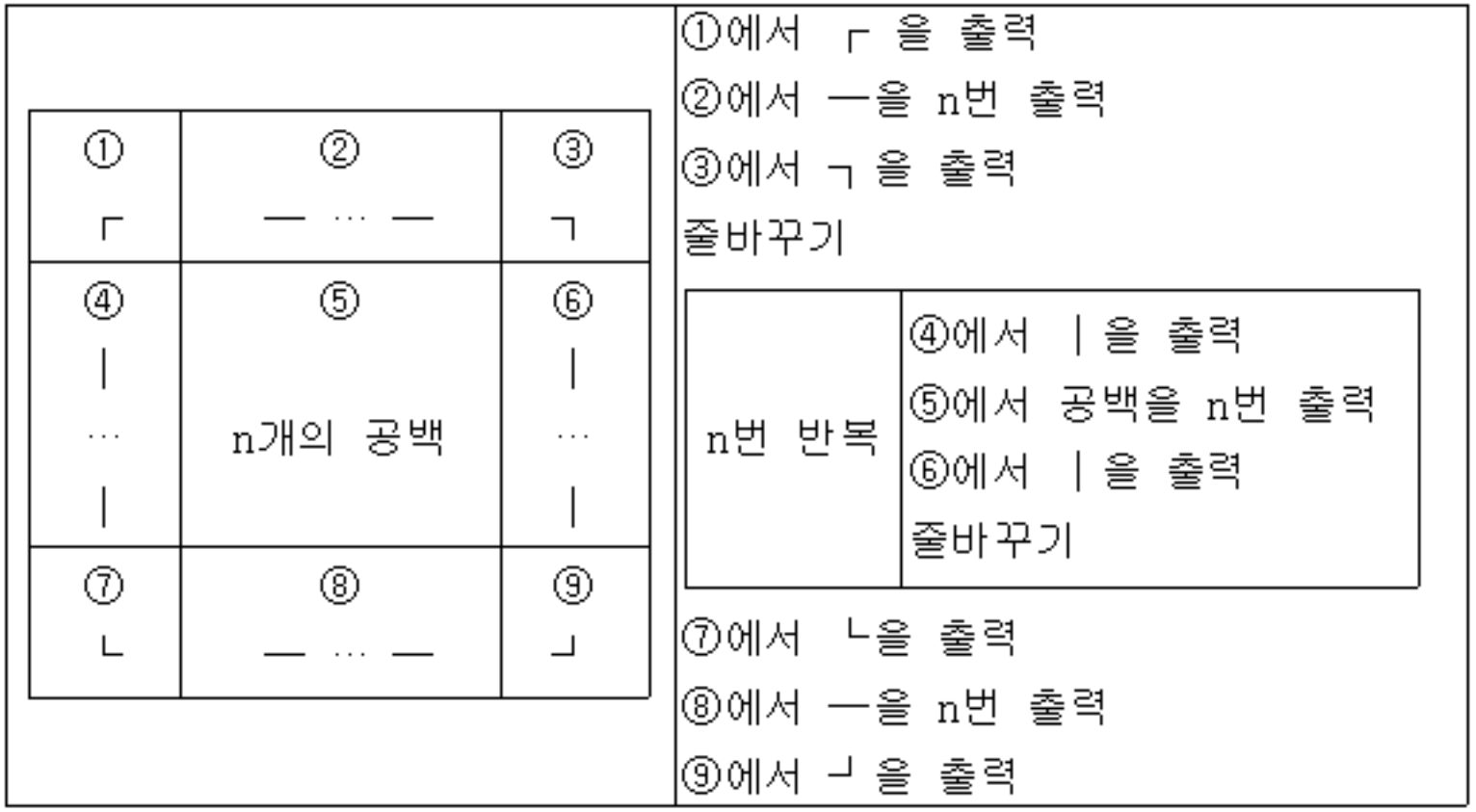
```
#include <stdio.h>
void draw_basic_square(void);

int main(void)
{
    draw_basic_square();
    return 0;
}

void draw_basic_square(void)
{
    unsigned char a=0xa6, b[7], i;
    for(i=1;i<7;i++)
            b[i]=0xa0+i;
    printf("%c%c", a, b[3]);
    printf("%c%c", a, b[4]);
    printf("₩n");
    printf("%c%c", a, b[6]);
    printf("%c%c", a, b[5]);
    printf("₩n");
}
```

길이가 n인 표준 정사각형

길이가 n인 표준 정사각형은 다음과 같이 반복적인 방법으로 정사각형의 크기를 표현 할 수 있다. 아래의 그림에서 숫자는 출력할 순서를 의미



```
#include <stdio.h>
void draw_square(int size);
int main(void)
{
    int n;
    printf("정사각형 그리기₩n₩n");
    printf("정사각형의 길이(최대 37)를₩n");
    printf("입력하고 Enter>");
    scanf("%d", &n);
    draw_square(n);
    return 0;
}
void draw_square(int size)
{
    int i, j;
    unsigned char a=0xa6;
    unsigned char b[7];
    for(i=1;i<7;i++)
        b[i]=0xa0+i;
```

```
 printf("%c%c",a, b[3]);
    for(i=0;i<size;i++)
        printf("%c%c", a, b[1]);
    printf("%c%c", a, b[4]);
    printf("₩n");
    for(i=0;i<size;i++)
    {
        printf("%c%c", a, b[2]);
        for(j=0;j<size;j++)
            printf("  ");
        printf("%c%c",a, b[2]);
        printf("₩n");
    }
    printf("%c%c", a, b[6]);
    for(i=0;i<size;i++)
        printf("%c%c", a, b[1]);
    printf("%c%c", a, b[5]);
    printf("₩n");
}
```

메뉴 만들기1

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int menu_display(void);
void hamburger(void);
void spaghetti(void);
void press_any_key(void);

int main(void)
{
 int c;
 while((c=menu_display()) != 3)
 {
  switch(c)
   {
    case 1 : hamburger();
                        break;
    case 2 : spaghetti();
                        break;
    default : break;
   }
 }
 return 0;
}
```

```
int menu_display(void)
{
        int select;
        system("cls");
        printf("간식 만들기₩n₩n");
        printf("1. 햄버거 ₩n");
        printf("2. 스파게티₩n");
        printf("3. 프로그램 종료₩n₩n");
        printf("메뉴번호 입력>");
        select=getch()-48;
        return select;
}
```

```
void hamburger(void)
{
        system("cls");
        printf("햄버거 만드는 방법₩n");
        printf("중략₩n");
        press_any_key();
}

void spaghetti(void)
{
        system("cls");
        printf("스파게티 만드는 방법₩n");
        printf("중략₩n");
        press_any_key();
}

void press_any_key(void)
{
 printf("₩n₩n");
 printf("아무키나 누르면 메인 메뉴로...");
 getch();
}`
```

메뉴 만들기2

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int menu_display(void);
int sub_menu_display01(void); //햄버거에 대한 서브 메뉴 출력과 번호 입력
int sub_menu_display02(void); //스파게티에 대한 서브 메뉴 출력과 번호 입력
void sub_main01(void); //햄버거에 대한 서브 메뉴 제어
void sub_main02(void); //스파게티에 대한 서브 메뉴 제어

void chicken_burger(void);
void cheese_burger(void);
void tomato_spaghetti(void);
void cream_spaghetti(void);
void press_any_key(void);  //아무키나 누르면 이전 메뉴로
```

```
int main(void)
{
    int c;
    while((c=menu_display())!=3)
    {
        switch(c)
        {
           case 1 : sub_main01();
                        break;
           case 2 : sub_main02();
                        break;
           default : break;
        }
    }
 return 0;
}
```

```
int menu_display(void)
{
    int select;
    system("cls");
    printf("간식 만들기₩n₩n");
    printf("1. 햄버거 ₩n");
    printf("2. 스파게티₩n");
    printf("3. 프로그램 종료₩n₩n");
    printf("메뉴번호 입력>");
    select=getch()-48;
    return select;
}
```

```
void sub_main01(void)
{
    int c;
    while((c=sub_menu_display01())!= 3)
    {
      switch(c)
      {
           case 1 : chicken_burger();
                                break;
           case 2 : cheese_burger();
                                break;
           default : break;
     }
   }
}
```

```
void chicken_burger(void)
{
           system("cls");
           printf("치킨버거 만드는 방법₩n");
           printf("중략₩n");
           press_any_key();
}
```

```
void cheese_burger(void)
{
           system("cls");
           printf("치즈버거 만드는 방법₩n");
           printf("중략₩n");
           press_any_key();
}
```

```
int sub_menu_display01(void)
{
    int select;
    system("cls");
    printf("햄버거 만들기₩n₩n");
    printf("1. 치킨버거₩n");
    printf("2. 치즈버거₩n");
    printf("3. 메인 메뉴로 이동₩n₩n");
    printf("메뉴번호 입력>");
    select=getch()-48;
    return select;
}
```

```
void sub_main02(void)
{
    int c;
    while((c=sub_menu_display02())!= 3)
    {
      switch(c)
      {
           case 1 : tomato_spaghetti();
                                break;
           case 2 : cream_spaghetti();
                                break;
           default : break;
     }
   }
}
```

```
int sub_menu_display02(void)
{
        int select;
        system("cls");
        printf("스파게티 만들기₩n₩n");
        printf("1. 토마토 스파게티 ₩n");
        printf("2. 크림 스파게티 ₩n");
        printf("3. 메인 메뉴로 이동₩n₩n");
        printf("메뉴번호 입력>");
        select=getch()-48;
        return select;
}
```

```
void cream_spaghetti(void)
{
        system("cls");
        printf("크림 스파게티 만드는 방법₩n");
        printf("중략₩n");
        press_any_key();
}
```

```
void tomato_spaghetti(void)
{
        system("cls");
        printf("토마토 스파게티 만드는 방법₩n");
        printf("중략₩n");
        press_any_key();
}
```

```
void press_any_key(void)
{
        printf("₩n₩n");
        printf("아무키나 누르면 이전 메뉴로...");
        getch();
}
```

범위 내의 난수 생성

로또복권과 같이 1부터 45사이 또는 0부터 99사이와 같이 특정한 범위 (구간)내에서의 난수를 생성하는 방법

특정한 범위내의 정수 난수를 생성하려면 함수 rand에 대해 아래 표와 같이 나머지 연산자 %와 덧셈 또는 뺄셈을 적절히 사용

정수 난수 생성범위	프로그램 연산식	설명
1 ≤ 정수난수 ≤ 6	rand()%6 + 1;	6으로 나눈 나머지 값의 범위는 0~5인데 이 값에 1을 더하므로 1~6사이의 난수 생성
1 ≤ 정수난수 ≤ 45	rand()%45 + 1;	45로 나눈 나머지 값의 범위는 0~44인데 이 값에 1을 더하므로 1~45사이의 난수 생성
0 ≤ 정수난수 ≤ 99	rand()%100;	100으로 나눈 나머지 값의 범위는 0~99이므로 0~99사이의 난수 생성
10 ≤ 정수난수 ≤ 30	rand()%21 + 10;	21로 나눈 나머지 값의 범위는 0~20인데 이 값에 10을 더해주면 10~30사이의 난수 생성
-5 ≤ 정수난수 ≤ 5	rand()%11 - 5;	11로 나눈 나머지 값의 범위는 0~10인데 이 값에 5를 감해주면 -5~+5사이의 난수 생성

주사위 눈금 난수 생성

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
 int i;
 srand(time(NULL));
 for(i=1;i<=10;i++)
    printf("%2d:%d₩n",i, rand()%6+1);
 return 0;
}
```

1부터 45 난수 생성

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
 int i;
 srand(time(NULL));
 for(i=1;i<=6;i++)
    printf("%2d:%d₩n",rand()%45+1);
 return 0;
}
```

1부터 45 중복 없는 난수 생성

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
   int i, j, lotto[6];
   srand(time(NULL));
   for(i=0;i<=5;i++)
   {
      lotto[i]=rand()%45+1;
      for(j=0;j<i;j++)
      {
                if (lotto[i] == lotto[j])
                {
                  i--;
                  break;
                }
      }
   }
   for(i=0;i<=5;i++)
              printf("%2d₩n", lotto[i]);
 return 0;
}
```

1부터 45 중복 없는 난수 생성 - 정렬

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void selection_sort(int r[], int n);
int main(void)
{
   int i, j, lotto[6];
   srand(time(NULL));
   for(i=0;i<=5;i++)
   {
      lotto[i]=rand()%45+1;
      for(j=0;j<i;j++)
      {
                if (lotto[i] == lotto[j])
                {
                  i--;
                  break;
                }
      }
   }
   selection_sort(lotto, 6);
   return 0;
}
```

```
void selection_sort(int r[], int n)
{
   int i, j, min, temp;
   for (i=0;i<=n;i++)
   {
        min = i;
        for (j=i+1;j<=n;j++)
                 if (r[j] < r[min])
                          min = j;
        temp = r[min];
        r[min] = r[i];
        r[i] = temp;
  }
  for(i=0;i<=5;i++)
                printf("%2d₩n", r[i]);
}
```

가변인수

- 가변 인수(variable argument)는 함수를 호출할 때 인수의 개수가 고정되어 있지 않고 변할 수 있는 인수
- 예를 들어 함수 printf를 사용할 때 값을 출력할 변수의 개수는 경우에 따라 달라질 수 있고, 함수 scanf의 경우에도 입력할 변수의 개수는 고정되어 있지 않다.

```
printf("%d", 34*15);  또는 printf("%d %d %d\n", a1, a2, a3);
scanf("%d", &input);  또는 scanf("%d %d %d", &a1, &a2, &a3);
```

printf	함수원형	`int printf(const char *format [, argument, ...]);`	
	함수인자	format	형식 제어 문자열(형식 지정자, 확장 문자)
		argument	변수나 상수 또는 연산식의 리스트
	반환 값	출력한 byte수를 반환하며, 오류 발생 시는 EOF를 반환. 반환 값은 거의 사용되지 않음	

- 가변 인수를 사용하는 함수의 원형에는 고정적으로 사용할 매개 변수가 최소한 한 개가 있어야 하고, 이후에 콤마와 ...를 함께 정의해야 한다.
- printf와 scanf의 경우에는 고정적으로 사용할 매개 변수는 한 개이고, 이는 형식 제어 문자열을 정의하는 부분
- 함수 원형에서 ...로 표시되는 가변 인수를 사용하려면 va_list라는 데이터 형을 이용하며 va_list 형은 헤더 파일 <stdarg.h>에 정의되어 있다.
- va_list 형은 가변 인수를 처리하는데 있어서 필요한 정보를 보관할 포인터 변수를 정의하기 위해 사용

```
#include <stdarg.h>
...
typedef char *  va_list;
...
void va_start(va_list ap, lastfix);
void va_arg(va_list ap, type);
void va_end(va_list ap);
```

- 가변 인수를 처리하는 함수를 정의할 때
  va_start, va_arg 그리고 va_end라는 매크로 함수를 사용.
  - 이 함수들은 인수의 개수와 데이터 형이 알려지지 않은 상태에서 함수가 호출되었을 경우 인수들을 처리하는데 사용
  - va_start는 두 개의 매개변수(ap와 lastfix)를 취합니다. 이 함수는 va_list 형 변수 ap를 초기화 하므로 함수 va_arg 또는 va_end을 호출하기 전에 사용
  - va_arg는 초기화가 완료된 va_list 형 변수 ap로부터 차례로 인수들을 반환하기위해 사용
  - va_end는 호출된 함수가 정상적인 반환을 수행할 수 있도록 도와주는 함수

가변 인수를 사용하는 함수의 기본적인 구조

```
데이터 형 function(고정된 매개 변수, ...)
{
   …
   va_list ap;
   va_start(ap, 고정된 매개 변수 중 마지막 매개 변수 이름);
   while(모든 인수를 차례로 다 읽었는가를 평가)
   {
      va_arg(ap, 인수의 데이터 형)에 의해 반환된 인수를 처리하는 부분
   }
   …
   va_end(ap);
}
```

```
#include <stdio.h>
#include <stdarg.h>
double sum(int count, ...);
int main(void)
{
   printf("합계 = %lf₩n", sum(2, 10.5, 20.23));
   printf("합계 = %lf₩n", sum(5, 10.3, 245.67, 0.51, 198345.764));
   return 0;
}

double sum(int count, ...)
{
   double total=0, number;
   int i=0;
   va_list ap;
   va_start(ap, count);
   while(i<count)
   {
     number=va_arg(ap, double);  //인수의 데이터 형은 double
     total+=number;
    i++;
   }
   va_end(ap);
   return total;
}
```

Reference

- ✓ 명품 C언어 프로젝트, 생능출판, 안기수
- ✓ http://egloos.zum.com/EireneHue/v/350618
- ✓ https://kcoder.tistory.com/entry/getchar-getch-getche%EC%9D%98-%EC%B0%A8%EC%9D%B4%EC%A0%90-%EC%98%88%EC%A0%9C%EC%86%8C%EC%8A%A4-%EA%B7%B8%EB%A6%BC
- ✓ https://plustag.tistory.com/1
- ✓ https://dojang.io/mod/page/view.php?id=763