◇ 리턴하지 않는 출력 함수

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ DESCRIPTION

C언어는 항상 main함수부터 컴파일이 시작된다. 그래서 바로 introduction함수를 호출한다고 보면 된다. introduction함수는 리턴형이 void로 아무 것도 반환하지 않는다. 여기서 반환이란, 호출된 함수에서 호출한 소스로 데이터를 보내는 것을 말한다. introduction이 호출되면 이름을 내보내는 출력문이 실행되며 프로그램이 종료된다.

◆ DESCRIPTION

◆ DESCRIPTION

#include <stdio.h>

void introduction()

{

 printf("Hi\nMy name is Chaitanya\nHow are you?");

}

int main()

{

 introduction();

}

◇ 함수와 배열로 로또 번호 출력

◆ DESCRIPTION

문제에서 1부터 45까지의 랜덤 수를 요구하였으니, 크기가 46인 정수형 배열을 선언하고 모든 요소를 0으로 초기화하였다. 그리고 count정수형 변수를 선언하고 1로 초기화하였다. 위 배열과 변수는 글로벌 변수이므로, 프로그램이 종료될 때까지 모든 함수에서 쓸 수 있는 변수이다. 프로그램은 기능은 block함수 내용이 전부이니, block만 설명하겠다. 반복문 내에서는 먼저 1부터 45까지 아무 수 한 개를 정수형 변수 i에 넣는다. 그리고 i를 인덱스로 가지는 배열 요소가 0이면 i를 출력한다. 기존 랜덤 수(i)는 이미 출력하였으니, i를 인덱스로 가지는 배열 요소 값을 1로 바꾼다. if문이 실행되야 count값이 올라가니, 모두 다른 랜덤 값 6개가 출력될 때까지 반복문은 멈추지 않는다.

#include <stdio.h>

# include <stdlib.h>

int x[46] = { 0 }, count = 1;

int block()

{

 srand(time(NULL));

 while (count <= 6) {

 int i = rand() % 45 + 1;

 if (x[i] == 0) {

 if (count == 6) {

 printf("%d", i);

 break;

 }

 printf("%d, ", i);

 x[i] = 1;

 count++;

 }

 }

}

int main(void)

{

 printf("생성된 로또 번호는 ");

 block();

 printf("입니다.");

}

◇ 리턴으로 원의 면적과 둘레 구하기

◆ DESCRIPTION

#include <stdio.h>

float area(float x)

{

 return 3.14\*x\*x;

}

float circum(float x)

{

 return 3.14 \* 2 \* x;

}

int main(void)

{

 float x = 0;

 while (x != -1)

 {

 printf("원의 반지름을 입력하시오:");

 scanf\_s("%f", &x);

 printf("원의 면적은 %f입니다.:\n", area(x));

 printf("원의 둘레는 %f입니다.:\n", circum(x));

 }

}

area함수는 실수형 데이터를 메개변수 x로 받고 실수형으로 리턴하는 함수이다.

리턴은 3.14\*x^2값을 보낸다.

circum함수는 실수형 데이터를 메개변수 x로 받고 실수형으로 리턴하는 함수이다. 리턴은 3.14\*2x값을 보낸다.

이 두 함수는 main함수에 의해서 호출된다. 실수형 변수 x를 먼저 선언하고 0으로 초기화하였다. 그리고 이 x값은 사용자의 입력으로부터 반지름으로 다시 정의된다. 이 x를 인수로 두고 함수를 호출하니 반지름에 따른 면적과 둘레를 리턴받아서 출력할 수 있는 것이다. 둘레,면적 구하기는 x가 -1이 아니면 계속 된다.

◇ 두 점 사이의 거리 구하기

◆ DESCRIPTION

두 점 사이의 거리를 구하려면 제곱근 함수가 필요하므로, 먼저 math.h헤더 파일을 불러왔다. main에서는 첫 번째 점과 두 번째 점의 좌표를 사용자 입력 값으로부터 가져와서 double형 정수 x1,x2,y1,y2에 저장하였다. 이 변수는 인수로 쓰여서 get\_distancce함수를 호출하게 된다. get\_distance는 double형 데이터 4개를 매개변수 x1,x2,y1,y2로 저장받고 double형으로 리턴하는 함수이다.

이 함수에서는 double형 변수 z를 선언 후, 매개변수를 이용한 거리 구하기 수식으로 초기화하였다.

이 z를 리턴하면서 두 점 사이의 거리를 구할 수 있게 된다.

#include <stdio.h>

#include <math.h>

double get\_distance(double x1, double x2, double y1, double y2)

{

 double z = sqrt((x1 - x2)\*(x1 - x2) + (y1 - y2)\*(y1 - y2));

 return z;

}

int main(void)

{

 double x1, x2, y1, y2;

 printf("첫 번째 점의 좌표를 입력하시오:(x,y)");

 scanf\_s("%lf%lf", &x1, &y1);

 printf("두 번째 점의 좌표를 입력하시오:(x,y)");

 scanf\_s("%lf%lf", &x2, &y2);

 printf("두 점 사이의 거리는%lf입니다.", get\_distance(x1, x2, y1, y2));

 return 0;

}

◇ 자동차 이동 게임

#include <stdio.h>

#include <windows.h>

HWND hwnd;

HDC hdc;

int main() {

 srand(time(NULL));

 hwnd = GetForegroundWindow();

 hdc = GetWindowDC(hwnd);

 int winner = 0; // 우승자 변수

 int minus = 0; // 기존 거리에서 뺀 값 변수

 int s1 = 800, s2 = 800, s3 = 800, gass = 0; // 차의 처음 위치 및 가스의 유무

 while (1) {

 setting();

 draw\_all(s1, s2, s3, gass);

 if (gass == 0) { // 게임 룰 설명 및 시작

 printf("게임을 시작하려면 아무 키나 입력해주세요!!");

 getch();

 printf("\n게임을 시작합니다.");

 printf("\n먼저 결승선을 통과한 차가 승리합니다. (배기구 기준)");

 Sleep(2000);

 printf("\n--------------");

 gass++;

 }

 if (s1 < -123 || s2 < -123 || s3 < -123) { // 배기구가 통과했을 때 우승자 정하기

 setting();

 draw\_all(s1, s2, s3, 0);

 if (s1 == s2 || s1 == s3 || s2 == s3)

 printf("비겼습니다!! 다시 시작해주세요!!");

 else if (s1 < -123 && s1 < s2 && s1 < s3)

 winner = 1;

 else if (s2 < -123 && s2 < s1 && s2 < s3)

 winner = 2;

 else if (s3 < -123 && s3 < s1 && s3 < s2)

 winner = 3;

 break;

 }

 minus = (rand() % 100) + 1; // 랜덤 값 지정

 s1 -= minus;

 minus = (rand() % 100) + 1;

 s2 -= minus;

 minus = (rand() % 100) + 1;

 s3 -= minus;

 Sleep(1000);

 }

 printf("%d번 자동차가 승리하였습니다!!!", winner);

◇ 자동차 이동 게임 함수

 printf("\n다시 시작하려면 Enter버튼을 클릭해주세요!!!", winner);

 getchar();

 main(); // 재귀함수를 이용해서 다시 시작!!

}

int draw\_all(int s1, int s2, int s3, int gass) { // 자동차 호출 함수

 draw(s1, 100, gass);

 draw(s2, 280, gass);

 draw(s3, 460, gass);

}

int setting(void) { // 세팅 함수

 Rectangle(hdc, 0, 0, 3000, 3000); // 배경 하얗게

 Rectangle(hdc, 100, 0, 133, 800); // 결승선 하양

 int one = 133, two = 166;

 for (int i = 0;i < 17;i++) {

 Rectangle(hdc, one, 0, two, 800); // 결승선 검정

 one++;two--;

 }

 Rectangle(hdc, 166, 0, 199, 800); // 결승선 하양

}

int draw(int x, int y, int gass) { // 자동차 그리기 함수

 Rectangle(hdc, x + 50, y - 50, x + 150, y); // 자동차 윗부분

 Rectangle(hdc, x, y, x + 200, y + 70); // 자동차 중심

 int a = y + 23, b = y + 48;

 for (int i = 0;i < 13;i++) {

 Rectangle(hdc, x, a, x + 200, b); // 중심 라인

 a++;b--;

 }

 Ellipse(hdc, x + 25, y + 60, x + 75, y + 110); // 자동차 앞 바퀴

 Ellipse(hdc, x + 125, y + 60, x + 175, y + 110); // 자동차 뒷 바퀴

 Rectangle(hdc, x + 200, y + 46, x + 223, y + 69); // 자동차 배기구

 if (gass == 1) {

 Ellipse(hdc, x + 228, y + 46, x + 238, y + 69); // 자동차 배기구 가스1

 Ellipse(hdc, x + 243, y + 41, x + 253, y + 74); // 자동차 배기구 가스2

 Ellipse(hdc, x + 258, y + 36, x + 268, y + 79); // 자동차 배기구 가스3

 }

}

◆ DESCRIPTION

= draw 함수 =

자동차를 그리는 함수이다. Rectangle함수를 이용해서 사각형을 만들고 Ellipse함수를 이용해서 원을 만들 수 있다. 위 함수를 호출할 때 두 번째 인수부터 네 번째 인수에는 좌표를 작성해주어야 하는데, (hdc, 기준 x좌표, 기준 y좌표, 끝점 x좌표, 끝점 y좌표)라고 생각하면 된다. 필자는 자동차 중심을 기준으로 해서 윗 부분과 바퀴를 만들었다. 즉 자동차 중심의 기준 좌표(x,y)들은 외부로 부터 받은 값(x,y)을 그대로 썼고, 윗 부분,바퀴,배기구 등은 (x+25,y+60) 등으로 작성해서 나타냈다고 보면 된다. 그리고 배기구 가스는 draw함수에서 gass매개변수가 1을 받을 때만 그려진다고 보면 된다. 주석 중에서 '중심 라인' 이란 것도 있는데 이건 자동차 중심에서 검은 색으로 되있는 부분을 말한다. 이 검은 색은 사각형을 계속 (자동차 중심 사각형) 안쪽에 만들어서 마치 검은 색 사각형인 듯 구현한 것이다. 기준 y좌표는 계속 키우고 끝점 y좌표는 계속 작게 하는 반복문을 이용하면 된다.

= setting 함수 =

배경과 결승선을 그리는 함수이다.

콘솔 창은 검은 색이기에 사각형이나 원을 만들면 (배경이 검정, 물체가 흰 색이라) 잘 안 보일 수 있다. 그래서 먼저 사각형을 엄청 크게 만들어서 마치 배경이 하얀 색인 것처럼 구현할 필요가 있다.

(그럼 배경이 하양, 물체가 검정이 됨) 그래서 처음에 엄청 큰 사각형을 만든 것이다. 이후 가로가 33,세로가 800인 사각형을 두 개 만들고 그 사이에 같은 크기의 검은 사각형을 만들었다. 검은 사각형을 만드는 법은 위 draw함수에서 중심 라인을 만들 때와 동일하다.

◆ DESCRIPTION

= draw\_all 함수 =

자동차를 호출하는 함수이다. 즉 draw함수를 호출하는 함수이다. 문제에서 자동차를 세 대 만들라고 하였으니 draw함수를 총 세 번 호출하였다. s1,s2,s3는 세 대의 자동차 각각의 (x좌표)위치이다. 그리고 뒤에 100,280,480은 각각의 (y좌표)위치이다. 여기서 자동차의 위치라고 했지만 엄밀히 따지자면 자동차 중심의 기준 x,y좌표가 되겠다. 결국 첫 번째 자동차가 차지하는 영역(y좌표 영역)은50(자동차 윗 부분 기준 y값)부터 210(자동차 바퀴 끝점 y값)이고, 두 번째 자동차(중심)를 첫 번째로부터 떨어놓아야 되니까 210(첫 번째 자동차 바퀴 끝점 y값)보다 약 20정도 더 떨어지게 하고 두 번째 자동차 윗 부분이 차지하는 y영역(50)을 고려해서 (자동차 중심 기준 y좌표)를 280으로 설정한 것이다. 즉 첫 번째 자동차와 두 번째 자동차 간의 떨어진 정도는 20이고, 첫 번째 자동차의 중심과 두번째 자동차의 중심 간의 떨어진 정도가 180이라고 생각하면 된다. 세 번째 자동차도 떨어진 정도는 같게 설정하였다. main함수에서 draw를 세 번 호출하는 것보다 draw\_all을 한 번 호출하는 것이 더 간단하다.

= main 함수 =

위의 함수들을 모두 호출하고, 게임 시작을 하고, 자동차의 위치를 변경하는 함수이다. 자동차 이동 게임이라고는 하나 사실 흰 배경을 초기 자동차 그림 위에 덮어 씌워 감추고, 다른 위치에 자동차를 그려서 마치 자동차가 이동한 것 처럼 보이는 방식이다. 매번 배경을 덮어 씌워야 되기에 draw\_all을 호출하기 이전에, 항상 setting함수를 호출해야 되는 것이다. draw\_all에는 자동차의 위치와 가스의 유무를 인수로 주어야 한다. (1을 주면 가스 나옴) 자동차의 위치(s1,s2,s3)는 반복을 할 때마다 달라지는데, s1-minus(1부터 100까지 나오는 랜덤 값), s2-minus(1부터 100까지 나오는 랜덤 값), s3-minus(1부터 100까지 나오는 랜덤 값)를 다시 s1,s2,s3에 저장하는 방식이어서 각각의 자동차 x좌표 위치가 매번 달라질 수밖에 없는 것이다. (이어서)

◆ DESCRIPTION

그리고 만약 자동차 배기구가 결승선을 통과할 때,[자동차 배기구 끝점 x값(sn+223)이 100(마지막 결승선의 기준 x값)보다 작을 때 => s1,s2,s3중 하나라도 -123보다 작을 때] s1,s2,s3값이 같은지 먼저 확인하고 같으면 비겼다고 알려주고 s1이 s2,s3보다 작으면 첫 번째 자동차가 이겼다고 알려준다. 나머지도 마찬가지다.

또한 결승선을 통과했다는 것은 자동차가 주행을 멈춰야 되니까 세팅을 한 후 가스를 0을 줘서 가스가 없는 자동차를 그린다. 이렇게 되면 게임이 끝나니, 사용자로부터 문자열을 입력받으면 main함수를 다시 호출하는 방식으로 재시작할 수 있게 만든다.

◆ Consequence



1. 게임 시작 화면



2. 아무 키나 눌러서 게임 시작 / 3. 자동차가 움직이면서 가스 배출



4. 한 대 이상 자동차의 배기구가 결승선을 통과하면 승리한 자동차를 알려주고 게임이 종료됨. 엔터 버튼을 클릭하여 다시 시작이 가능함!!