C프로그래밍 실습

아스키 아트 생성 프로그램

최종 보고서

제출일자:23/12/20
제출자명:김진강
제출자 학번:233575

1. 프로젝트 목표

1) 배경 및 필요성

좁은 의미로서의 아스키 아트(ASCII Art) 란 아스키 코드 0x20~0x7e에 포함되는 문자, 기호를 사용한 그림을 말함. 요즘은 1바이트(8비트)의 문자 이외에, 2바이트(16비트)의 문자 같이 정확히 말하면 아스키 코드에 포함되지 않는 문자도 아스키아트라는 용어로 불리며, 이를 활용하여 더욱 다양한 아스키아트가 만들어지고 있음. 이러한 아스키코드는 사용자가 수작업으로 만드는데 어려움이 있음. 이러한 문제를 아스키아트를 자동으로 생성해주는 프로그램으로 해결할 수 있음.

2) 프로젝트 목표

사용자의 이미지를 사용자가 원하는 문자로 구성된 아스키아트로 만들어주는 프로그램.

3) 차별점

기존 프로그램은 정해진 문자로만 아스키아트를 만들어 줌. 이는 다양한 아스키아트의 수요에 맞추지 못할 수 있음. 우리는 아스키아트를 생성할 때 문자를 사용자가 직접 선택할 수 있게 하여 차별점이 있으며 사용자가 사이즈를 직접 지정할 수 있음.

2. 기능 계획

1) 사용자가 사용할 문자를 설정하는 기능

- 기본으로 설정된 문자 8개에서 사용자가 사용할 문자열을 변경할 수 있게 함.

(1) 사용할 문자는 8개로 배열에 명암이 오름차순으로 저장됨

2) 사용자의 이미지를 불러오는 기능

- 사용자에게 파일 이름을 입력 받아서 파일을 불러옴

- libjpeg 라이브러리를 활용하여 jpg이미지를 불러오고 처리한다.

(1) 파일 형식은 jpeg로 고정

(2) 이미지의 픽셀을 몇 사이즈씩 평균 낼 지 사용자에게 입력받음

3) 이미지를 아스키아트로 변환하는 기능

- 이미지를 흑백으로 변환하고 명암을 구한 후 그 명암에 맞는 글자를 출력함.

- libjpeg 라이브러리를 활용하여 jpg이미지를 불러오고 처리한다.

(1) 이미지의 RGB값을 흑백으로 변환함

(2)명암에 따라 1)에서 저장한 문자를 선택하여 출력

3. 기능 구현

(1) 사용자의 이미지를 불러오는 기능

- 사용자가 입력한 파일 불러옴

- 1) fopen 함수를 사용하여 지정된 JPEG 파일을 이진 읽기 모드("rb")로 연다.(안 열리면 메시지 출력 후 종료)

2) jpg 구조체를 선언

3) 오류 처리기 설정. jpg파일에 문제가 있는 경우 에러를 출력하고 종료

4) jpeg_stdio_src 함수를 사용하여 JPEG 디코딩에 사용할 입력 파일을 설정

5) jpeg_read_header 함수를 호출하여 JPEG 파일의 헤더 정보를 읽음

6) jpeg_start_decompress 함수를 호출하여 이미지 디코딩을 시작

7) 이미지 데이터를 저장할 버퍼를 할당하고 (malloc 함수 사용), jpeg_read_scanlines 함수를 사용하여 행 단위로 이미지 데이터를 읽음.(이미지를 아래에서 위로 읽는다.)

8) 명암에 따라 1)에서 저장한 문자를 선택하여 출력

- 적용된 배운 내용

파일 입출력(1), 반복문(7), 배열(7), 포인터(6), 구조체, 동적 메모리(8)

- 코드 스크린샷

```
// JPEG 파일을 읽기 모드로 열기
FILE* file = fopen(info.filename, "rb");
if (!file) {
    printf("파일을 열 수 없습니다.");
    return 1;
}

// JPEG 구조체 선언
struct jpeg_decompress_struct cinfo;
struct jpeg_error_mgr jerr;

// JPEG 오류 처리기 설정
cinfo.err = jpeg_std_error(&jerr);

// JPEG 구조체 초기화
jpeg_create_decompress(&cinfo);

// 입력 파일 지정
jpeg_stdio_src(&cinfo, file);

// 파일의 헤더 정보 읽기
jpeg_read_header(&cinfo, TRUE);

// 이미지 디코딩
jpeg_start_decompress(&cinfo);

// 이미지 데이터 저장을 위한 버퍼 생성
unsigned char* image = (unsigned char*)malloc(cinfo.output_width * cinfo.output_height * cinfo.output_components);

// 이미지 행 단위로 스캔하며 데이터 읽기
while (cinfo.output_scanline < cinfo.output_height) {
    unsigned char* row_pointer = &image[(cinfo.output_height - cinfo.output_scanline - 1) * cinfo.output_width * cinfo.output_components];
    jpeg_read_scanlines(&cinfo, &row_pointer, 1);
}
```

```
//사이즈 설정
while (1)
{
    printf("몇 곱하기 몇 픽셀 당 하나의 문자를 할당할 것인지 입력해주세요(숫자 하나만 입력): ");
    scanf_s("%d", &info.size);
    if (info.size > cinfo.output_width || info.size > cinfo.output_height) {
        printf("사진의 크기보다 클 수 없습니다.₩n");
    }
    else {
        break;
    }
}
```

(2) 이미지를 아스키아트로 변환하는 기능

- 입력: 앞에서 값을 저장한 "image"변수, 불러온 이미지의 가로 길이, 불러온 이미지의 세로 길이

- 2중 for 문에서 행은 뒤에서 앞으로 읽어온다.(libpeg는 이미지를 아래에서 위로 읽으므로) 픽셀을 하나씩 흑백으로 변환하고 명암을 구한 후 그 명암에 맞는 글자를 출력하고 행 for문이 한번 돌때마다 줄바꿈함.

- 적용된 배운 내용

반복문, 함수, 포인터, 구조체

```
// 이미지 출력
changeToAscii(image, cinfo.output_width, cinfo.output_height);
```

```
struct ImageInfo info;
strcpy_s(info.ascii_chars, sizeof(info.ascii_chars), "@%#*+=-:. ");
char changeAscii;
int charIndex;
char changeAsciiword;
while (1)
{
    printf("아스키 아트 생성에 사용할 문자를 변경하시겠습니까?(Y/N)₩n현재 문자(%s) ", info.ascii_chars, sizeof(info.ascii_chars));
    scanf_s(" %c", &changeAscii,sizeof(changeAscii));
    if (changeAscii == 'Y' || changeAscii == 'y') {
        printf("변경할 문자의 번호를 입력하세요.₩n현재 문자(%s) ", info.ascii_chars, sizeof(info.ascii_chars));
        scanf_s("%d", &charIndex);
        printf("변경할 문자를 입력하세요.₩n현재 문자(%s) ", info.ascii_chars, sizeof(info.ascii_chars));
        scanf_s(" %c", &changeAsciiword,sizeof(changeAsciiword));
        info.ascii_chars[charIndex - 1] = changeAsciiword;
    }
    else if (changeAscii == 'N' || changeAscii == 'n') {
        break;
    }
    else
    {
        printf("Y 또는 N만 입력해주세요.");
    }

    // 개행 문자 처리
    while (getchar() != '₩n');
}

printf("jpeg 파일의 경로를 입력해주세요. ");
scanf_s("%s", info.filename, sizeof(info.filename));
```

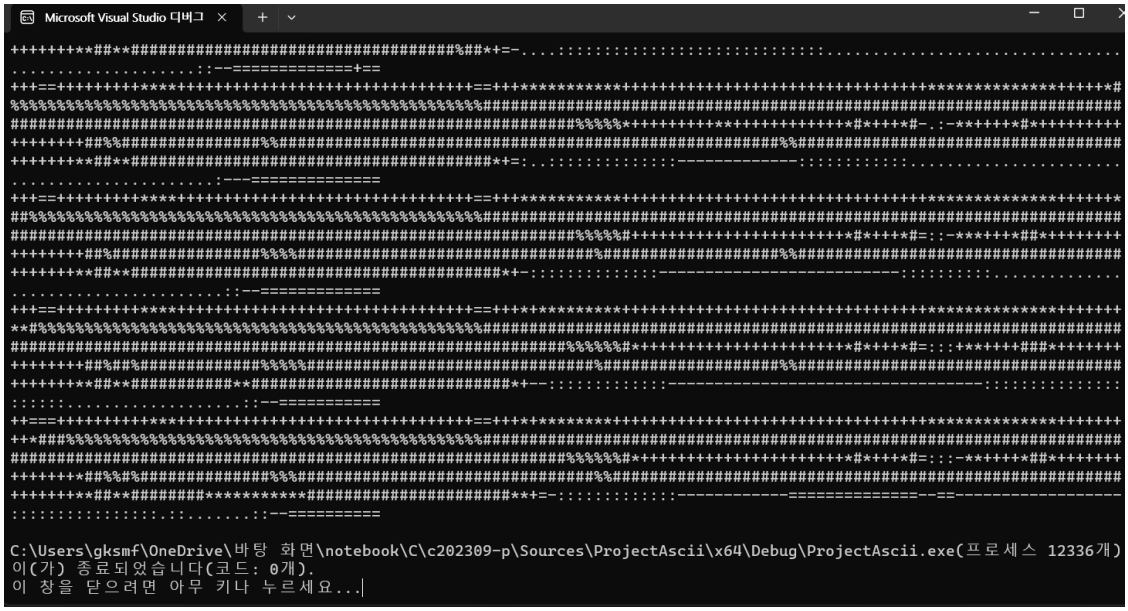
2) 테스트 결과

(1) 사용자의 이미지를 불러오는 기능과 이미지를 아스키아트로 변환하는 기능

- 1) 사용자가 입력한 파일을 불러옴

2) 2중 for 문에서 행은 뒤에서 앞으로 읽어온다.(libpeg 라이브러리는 이미지를 아래에서 위로 읽으므로) 픽셀을 하나씩 흑백으로 변환하고 명암을 구한 후 그 명암에 맞는 글자를 출력하고 행 for문이 한번 돌때마다 줄 바꿈.

- 테스트 결과 스크린샷(이미지 크기는 640x480로 픽셀 하나에 한 문자를 할당해서 한 화면에 담기지 않음)(1x1)



원본 이미지

메모장에 따로 복사한 사진(1x1)

- 테스트 결과 스크린샷(변경 후)(10x10으로 평균 냄)

3) 사용자에게 사용할 문자와 파일 경로를 입력받는 기능

입력) 변경 여부, 사용할 문자, 파일 경로

-사용자에게 사용할 문자를 바꿀지 물어보고 바꿀 문자의 번호를 물어본 후 사용할 문자

를 물어봄. 문자 변경이 완료되면 파일 경로를 입력 받음

-테스트 결과 스크린샷

4. 계획 대비 변경 사항

1) 픽셀 수 변경

- 이전: 한 픽셀에 하나씩 문자를 할당

- 이후: 여러 픽셀의 명암을 평균내서 하나의 문자를 할당

- 사유: 한눈에 보기 힘들고 글자수가 너무 많으면 활용도가 떨어짐

1) 평균 낼 픽셀의 크기를 사용자가 입력

- 이전: 10x10 픽셀에 하나씩 문자를 할당
- 이후: 사용자가 평균 낼 픽셀의 크기 정의
- 사유: 사용자가 원하는 아스키 아트를 출력해줄 수 있음

6. 느낀 점

1학기에 들었던 코딩 교양 과목보다 내용이 자세해서 훨씬 배우는 게 많았던 것 같습니다. 수업 난이도도 다른 학생들은 어렵다고 하는 학생도 있지만 저는 오히려 너무 쉬우면 배워가는 게 별로 없다고 생각해서 딱 적당한 난이도였다고 생각합니다. 그리고 평소 요즘 인터넷 없이 코딩하는 사람이 없으니까 코딩 시험도 오픈북이 났지 않을까 생각하곤 했는데 시험을 오픈북 형식으로 해서 아주 합리적이고 좋았습니다. 깃 헙이나 실무 관련된 언급도 강의중에 계속 해주셔서 얻어가는 게 더 많았던 것 같습니다.