8장 파일 입출력

2023년도 1학기 윈도우 프로그래밍

학습 목표

- 학습목표
  - API에서 제공하는 파일 입출력을 배운다.
  - 공용 대화상자 사용법에 대하여 배운다.

- 내용
  - 파일 다루기
  - 공용 대화상자

1. 파일 다루기

- API 이용한 표준 입출력 및 파일 사용 방법
  - 파일을 만들고 열어준다. 열 때는 읽기용인지 쓰기용인지 명시한다.
  - 열린 파일에는 텍스트를 쓰거나 읽는다.
  - 작업 후에는 파일을 닫는다.

기능	C언어 표준 라이브러리 함수	Win32 API함수
파일 열기	fopen()	CreateFile()
파일 닫기	fclose()	CloseHandle()
파일 포인터 위치변경/획득	fseek()	SetFilePointer()
파일 읽기	fread()	ReadFile()
파일 쓰기	fwrite()	WriteFile()

파일 생성/열기 함수

- 파일 생성 함수

```
HANDLE CreateFile  (LPCTSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode,
                    LPSECURITY_ATTRIBUTES lpSecurityAttributes, DWORD dwCreationDisposition,
                    DWORD dwFlagsAndAttributes, HANDLE hTemplateFile);
```

- 파일을 생성하거나 열 수 있다.
  - lpFileName: 만들 파일 이름
  - dwDesiredAccess: 읽기/쓰기 모드 (아래의 3 모드 중 1개 지정)
    - 읽기: **GENERIC_READ**
    - 쓰기: **GENERIC_WRITE**
    - 읽기 및 쓰기: **GENERIC_READ| GENERIC_WRITE**
  - dwShareMode: 공유 모드 (파일 공유 여부 명시)
    - 읽기 공유 허용: **FILE_SHARE_READ**
    - 쓰기 공유 허용: **FILE_SHARE_WRITE**
  - lpSecurityAttributes: 보안 속성 (자녀 프로세스에 상속 여부 설정), NULL 이면 상속 안됨
  - dwCreationDistribution:파일 생성 모드
    - 새로 만들기, 이미 있으면 에러 메시지: **CREATE_NEW**
    - 항상 새로 만들기, 파일이 있어도 파괴하고 새로 만듦: **CREATE_ALWAYS**
    - 기존 파일 열기, 파일이 없으면 에러 메시지: **OPEN_EXISTING**
    - 항상 열기: **OPEN_ALWAYS**
  - dwFlagsAndAttributes: 파일 속성 (읽기 전용 파일, 시스템 파일, 숨겨진 파일 등 지정)
    - 일반적인 파일: **FILE_ATTRIBUTE_NORMAL**
  - nTemplateFile: 생성될 파일의 속성을 제공할 템플릿 파일

파일 생성/열기 함수

- 파일 읽기

```
BOOL ReadFile (HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToRead,
               LPDWORD lpNumberOfBytesRead, LPOVERLAPPED lpOverlapped);
```

  - 파일 읽기 함수
    - hFile: 데이터를 읽을 파일 핸들
    - lpBuffer: 읽은 자료를 넣을 버퍼
    - nNumberOfBytesToRead: 읽고자 하는 바이트 크기
    - lpNumberOfBytesRead: 실제 읽은 바이트
    - lpOverlapped: NULL

- 파일 쓰기

```
BOOL WriteFile (HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToWrite,
                LPDWORD lpNumberOfBytesWritten, LPOVERLAPPED lpOverlapped);
```

  - 파일 쓰기 함수
    - hFile: 데이터를 저장할 파일 핸들
    - lpBuffer: 쓸 자료가 저장된 버퍼
    - nNumberOfBytesToWrite : 쓸 자료의 바이트 크기
    - lpNumberOfBytesWritten : 실제 쓴 바이트
    - lpOverlapped: NULL

- 파일 닫기

```
void CloseFile (HANDLE hFile);
```

  - 파일 닫기 함수
    - hFile: 닫을 파일 핸들

임의 접근

- 파일 엑세스할 때 대상 파일 위치 (File Pointer) 결정
  - 최초로 파일이 열렸을 때: FP는 파일의 선두 위치, 파일을 읽거나 쓰면 그만큼 파일 포인터가 이동 → 순차적 접근
  - 파일의 임의의 위치에서 원하는 만큼 읽는다. → 임의 접근

```
DWORD SetFilePointer  (HANDLE hFile, LONG lDistanceToMove,
                       PLONG lpDistanceToMoveHight, DWORD dwMoveMethod);
```

- 임의 접근 함수
  - hFile: 파일 핸들
  - lDistanceToMove: 파일 포인터가 이동할 위치
  - lpDistanceToMoveHight: 파일 크기가 2GB 이상일 경우 옮길 위치
  - dwMoveMethod: 파일 포인터의 이동 시작 위치 지정
    - FILE_BEGIN / FILE_CURRENT / FILE_END

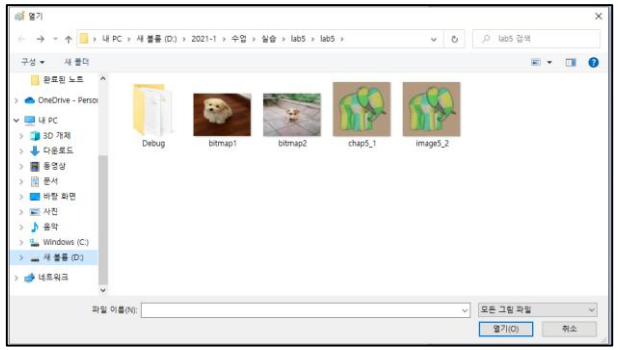
파일 입출력 예

- 사용 예) 기존 텍스트파일을 열어 화면에 출력하고, 데이터를 파일에 저장하기

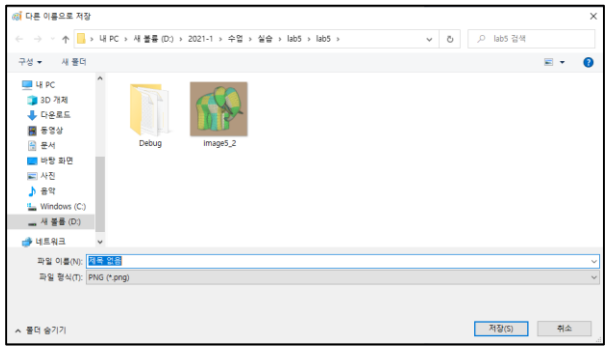
```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
	HDC hdc;
	HANDLE hFile;
	TCHAR InBuff[1000];
	TCHAR OutBuff[1000] = L"API 파일 입출력 테스트입니다.";
	int size = 1000, read_size;

	switch (iMsg) {
	case WM_LBUTTONDOWN:
		hFile = CreateFile ( L"test.txt", GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, 0);
		memset (InBuf, 0, sizeof (InBuff));
		ReadFile (hFile, InBuff, size, &read_size, NULL);                    //--- hFile에서 size 만큼 읽어 InBuff에 저장
		InBuff[read_size] = '₩0';

		hdc = GetDC(hwnd);
		TextOut (hdc, 0, 0, InBuff, lstrlen(InBuff));                        //--- InBuff 에 있는 내용을 화면에 출력
		ReleaseDC (hwnd, hdc);

		SetFilePointer (hFile, 0, NULL, FILE_END);
		WriteFile (hFile, OutBuff, lstrlen(OutBuff)*sizeof(TCHAR), &size, NULL);   //--- OutBuff의 내용을 hFile의 끝에 저장
		CloseHandle (hFile);
	break;
}
```

2. 공용대화상자

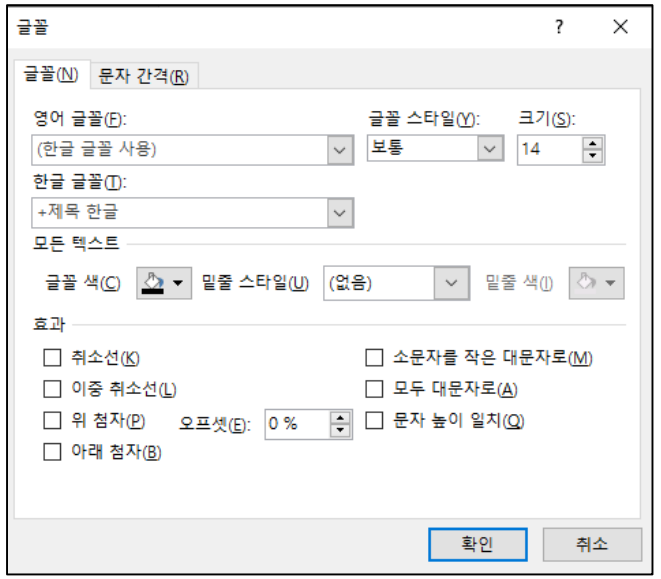
- 윈도우의 공용 대화상자
  - 파일 열기, 글꼴 선택하기, 색상 선택하기 등의 윈도우
  - 공용 대화상자 사용하기
    - 해당 대화 상자의 구조체 설정 → 함수 호출



파일 열기



파일 저장하기



글꼴 선택하기

1) 공용대화상자: 파일 열기

- 파일열기 처리절차
  - **OPENFILENAME** 구조체 할당
  - 공용대화상자 열기함수 호출

OPENFILENAME 구조체

```
typedef struct tagOFN{                                  //--- ofn
    DWORD   lStructSize;                                //--- 구조체 크기
    HWND                hwndOwner;                      //--- 오너 윈도우 핸들
    HINSTANCE           hInstance;                      //--- 인스턴스 핸들
    LPCTSTR             lpstrFilter;                    //--- 파일 형식 콤보 박스에 나타낼 필터
    LPTSTR              lpstrCustomFilter;              //--- 커스텀 필터를 저장하기 위한 버퍼
    DWORD               nMaxCustFilter;                 //--- 커스텀 필터 버퍼의 길이
    DWORD               nFilterIndex;                   //--- 파일 형식 콤보 박스에서 사용할 필터의 인덱스
    LPTSTR              lpstrFile;                      //--- 파일 이름 에디트에 처음 나타낼 파일명
                                                        //--- 최종적으로 선택된 파일이름이 저장된다.
    DWORD               nMaxFile;                       //--- lpstrFile 멤버의 길이
    LPTSTR              lpstrFileTitle;                 //--- 선택한 파일명을 리턴받기 위한 버퍼 (경로X)
    DWORD               nMaxFileTitle;                  //--- lpstrFileTitle 멤버의 길이
    LPCTSTR             lpstrInitialDir;                //--- 파일 찾기를 시작할 디렉토리
    LPCTSTR             lpstrTitle;                     //--- 대화상자의 캡션
    DWORD               Flags;                          //--- 대화상자의 모양과 동작을 지정하는 플래그
    WORD                nFileOffset;                    //--- lpstrFile 버퍼 내의 파일명 오프셋
    WORD                nFileExtension;                 //--- lpstrFile 버퍼 내의 파일 확장자 오프셋
    LPCTSTR             lpstrDefExt;                    //--- 디폴트 확장자
    DWORD               lCustData;                      //--- 훅 프로시저로 보낼 사용자 정의 데이터
    LPOFNHOOKPROC       lpfnHook;                       //--- 훅 프로시저명
    LPCTSTR             lpTemplateName;                 //--- 템플리트명
} OPENFILENAME;
```

파일 공용 대화상자

- 파일 공용 대화상자를 열기 위한 함수

BOOL **GetOpenFileName** (LPOPENFILENAME lpofn);
- 파일 입출력을 위해 파일 공용 대화상자를 열어 대상 파일을 입력받기 위해 호출되는 함수
- lpofn: **입력**을 위한 OPENFILENAME 구조체 포인터

BOOL **GetSaveFileName** (LPOPENFILENAME lpsfn);
- 파일 입출력을 위해 파일 공용 대화상자를 열어 대상 파일을 입력받기 위해 호출되는 함수
- lpofn: **출력**을 위한 OPENFILENAME 구조체 포인터

파일 열기

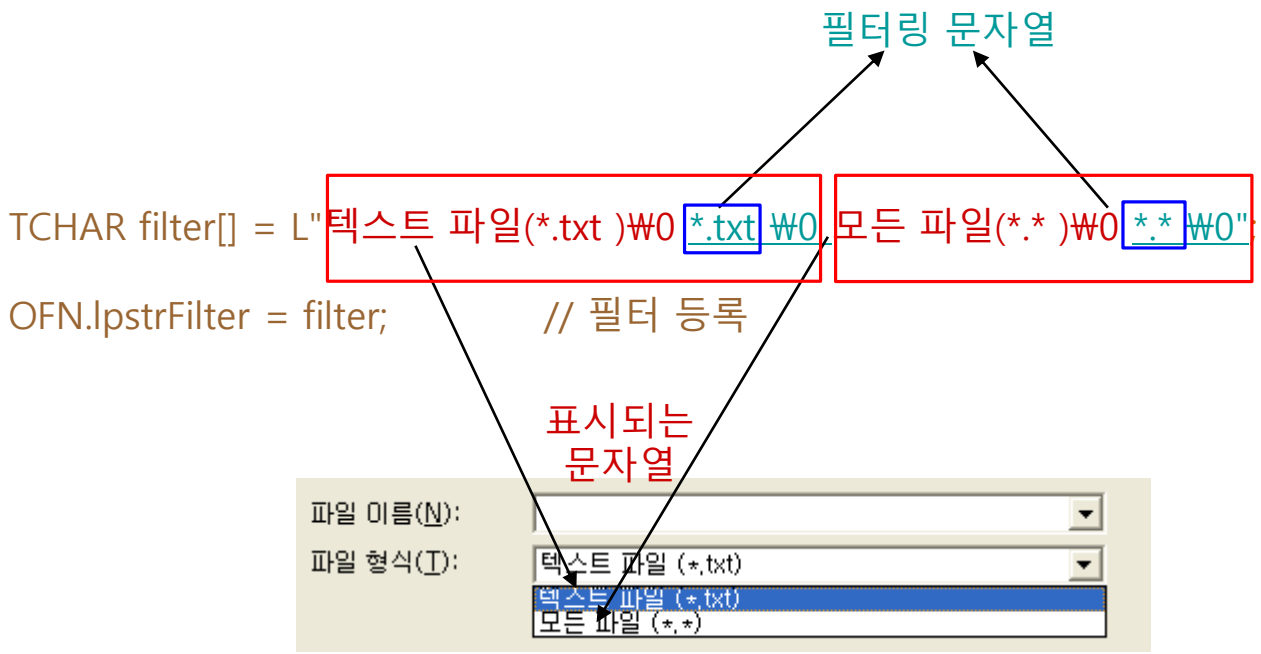
- 파일열기 처리절차
  - **OPENFILENAME** 구조체 할당
  - 열기함수 호출 → 파일이름 획득

```
OPENFILENAME OFN;                                   //--- 구조체 선언

memset (&OFN, 0, sizeof(OPENFILENAME));             //--- 구조체 초기화
OFN.lStructSize = sizeof(OPENFILENAME);
OFN.hwndOwner = hwnd;
OFN.lpstrFilter = filter;
OFN.lpstrFile = lpstrFile;
OFN.nMaxFile = 256;
OFN.lpstrInitialDir =  " . " ;
if (GetOpenFileName (&OFN)!=0) {                    //--- 파일 함수 호출
        wsprintf (str, "%s 파일을 여시겠습니까 ?", OFN.lpstrFile);
        MessageBox (hwnd, str, L"저장하기 선택", MB_OK);
}
```

void **memset** (void *ptr, int value, size_t num);
- 어떤 메모리의 시작점부터 연속된 범위를 어떤 값으로 모두 지정할 때 사용
  - ptr: 채우고자 하는 메모리의 시작 포인터 (주소값)
  - value: 메모리에 채우고자 하는 값
  - num: 채우고자 하는 바이트의 수 (메모리 크기)

필터 지정 방법

- 필터의 용도
  - 표시되는 파일이름을 걸러 줌
  - 정의시 공문자 삽입 안 하도록
  - 매 필터마다 널 문자로 종료하며 하나의 필터는 **"파일형식₩0필터₩0"**로 표시한다.
  - 한 개의 필터에 여러 개의 패턴 지정하려면 **; (세미콜른)**으로 연결



파일 열기

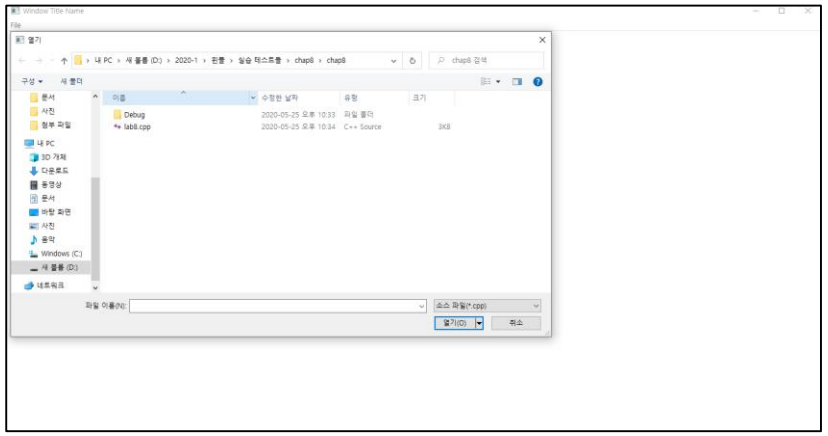
- 사용 예) 파일 열기 함수 사용하기
  - 메뉴를 사용하여 파일 열기 공용 대화상자 열기

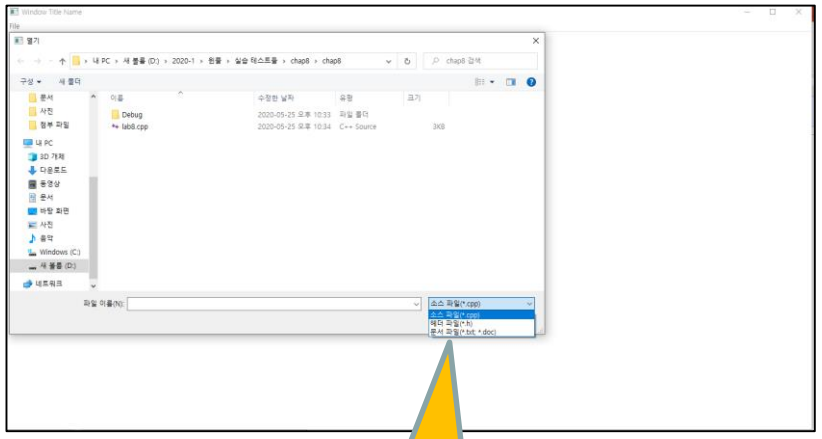
```
OPENFILENAME OFN;                                   //--- 파일 관련 구조체 선언
TCHAR str[100], lpstrFile[100] = L"";
TCHAR filter[100] = L"소스 파일(*.cpp)₩0*.cpp₩0헤더 파일(*.h)₩0*.h₩0문서 파일(*.txt; *.doc) ₩0*.txt;*.doc₩0";

switch (iMsg)
{
case WM_COMMAND:
        switch(LOWORD(wParam)) {
           case ID_FILEOPEN:                                //--- 메뉴 선택
           memset(&OFN, 0, sizeof(OPENFILENAME));           //--- 구조체 초기화
           OFN.lStructSize = sizeof(OPENFILENAME);
           OFN.hwndOwner = hwnd;
           OFN.lpstrFilter = filter;
           OFN.lpstrFile = lpstrFile;
           OFN.nMaxFile = 256;
           OFN.lpstrInitialDir = L".";                      //--- 초기 디렉토리

           if (GetOpenFileName (&OFN)!=0) {                 //--- 파일 열기 함수 호출
                wsprintf (str, L"%s 파일을 여시겠습니까 ?", OFN.lpstrFile);
              MessageBox (hwnd, str, L"열기 선택", MB_OK);
           }
           break;
```

*.txt와 *.doc 파일

파일 열기

- 결과 화면





필터 설정: *.cpp
*.h
*.hwp *.doc

파일 저장하기

- 사용 예) 파일 저장하기 함수 사용하기
  - 메뉴를 사용하여 파일 저장하기 공용 대화상자 열기

```
OPENFILENAME SFN;                                          //--- 파일열기와 저장하기는 동일한 구조체 사용
TCHAR str[100], lpstrFile[100] = L"";
TCHAR filter[100] = L"소스 파일(*.cpp)₩0*.cpp₩0헤더 파일(*.h)₩0*.h₩0문서 파일(*.txt; *.doc) ₩0*.txt;*.doc₩0";

switch (iMsg)
{
case WM_COMMAND:
      switch(LOWORD(wParam)) {
         case ID_FILESAVE:                                                 //--- 메뉴 선택
            memset (&OFN, 0, sizeof(OPENFILENAME));                        //--- 초기화
            SFN.lStructSize = sizeof(OPENFILENAME);
            SFN.hwndOwner = hwnd;
            SFN.lpstrFilter = filter;
            SFN.lpstrFile = lpstrFile;
            SFN.nMaxFile = 256;
            SFN.lpstrInitialDir = ".";

            if (GetSaveFileName (&SFN)!=0) {                               //--- 파일에 저장하기 함수 호출
                  wsprintf (str, L"%s 파일에 저장하시겠습니까 ?", SFN.lpstrFile);
                  MessageBox (hwnd, str, L " 저장하기 선택", MB_OK);
            }
            break;
```

2) 공용 대화상자: 폰트 선택하기

- 폰트 선택하기 처리절차
  - CHOOSEFONT 구조체 할당
  - LOGFONT 구조체 변수 연결
  - 폰트대화상자 띄우기 → 폰트정보 획득
  - 폰트 만들어 사용하기

```
CHOOSEFONT FONT;
LOGFONT    LogFont;

FONT.lStructSize = sizeof(CHOOSEFONT);          //--- 구조체 크기
FONT.hwndOwner = hwnd;                          //--- 윈도우 핸들
FONT.lpLogFont= &LogFont;                       //--- LOGFONT 구조체 변수 연결
FONT.Flags = CF_EFFECTS | CF_SCREENFONTS;       //--- 폰트대화상자 옵션
ChooseFont (&FONT)                              //--- 폰트대화상자 띄우기

hFont = CreateFontIndirect(&LogFont);           //--- 선택된 폰트 핸들 생성
OldFont = (HFONT)SelectObject(hdc, hFont);      //--- 폰트 사용
```

폰트 선택하기

- CHOOSEFONT 구조체

```
typedef struct {
        DWORD       lStructSize;          //---  구조체 크기
        HWND        hwndOwner;            //--- 메인윈도우 핸들
        HDC         hDC;                  //---  메인 DC 핸들
        LPLOGFONT   lpLogFont;            //---  LOGFONT 구조체 변수 값
                                          //--- (글꼴 선택하면 설정된다.)
        int         iPointSize;           //---  선택한 글꼴의 크기 (글꼴 선택하면 설정된다)
        DWORD       Flags;                //---  글꼴 상자 초기화
        COLORREF    rgbColors;            //---  선택한 글꼴의 색상 정보 저장
        LPARAM      lCustData;
        LPCFHOOKPROC lpfnHook;
        LPCTSTR     lpTemplateName;
        HINSTANCE   hInstance;
        LPSTR       lpszStyle;
        WORLD       nFontType;            //---  선택한 글꼴을 가리키는 필드
        int         nSizeMin;
        int         nSizeMax;
} CHOOSEFONT, *LPCHOOSEFONT;
```

COLORREF **SetTextColor** (HDC hdc, COLORREF crColor);

디바이스 컨텍스트에 이미 등록되어 있던 텍스트 색상값을 반환

- hdc: 변경할 디바이스 컨텍스트
- Color: 변경할 색상

폰트 선택하기

- LOGFONT 구조체

```
typedef struct {
        LONG        lfHeight;                   //--- 논리적 크기의 글꼴의 높이를 나타내는 정수
        LONG        lfWidth;                    //--- 글꼴의 너비
        LONG        lfEscapement;
        LONG        lfOrientation;
        LONG        lfWeight;                   //--- 글꼴의 굵기 지정 (0 ~ 100 사이의 정수)
        BYTE        lfIatlic;                   //--- 이탤릭 체 (TRUE/FALSE)
        BYTE        lfUnderline;                //--- 글자에 밑줄 (TRUE/FALSE)
        BYTE        lfStrikeOut;                //--- 글자에 취소선 (TRUE/FALSE)
        BYTe        lfCharSet;
        BYTE        lfOutPrecision;
        BYTE        lfQuality;
        BYTE        lfPitchAndFamily;
        TCHAR       lfFaceName[LF_FACESIZE];    //---  문자 배열로 글꼴 이름 저장
} LOGFONT;
```

폰트 선택하기

- 폰트 다루기 함수들

HFONT **CreateFont** (int nHeight, int nWidth,int nEscapement, int nOrientation, int fnWeight, DWORD fdwItalic, DWORDfdwUnderline, DWORD fdwStrikeOut, DWORD fdwCharSet, DWORD fdwOutputPrecision,DWORD fdwClipPrecision, DWORD fdwQuality, DWORD fdwPitchAndFamily, LPCTSTRlpszFace);

- 인수가 지정하는 특성에 가장 일치하는 논리 폰트를 생성하는 함수

HFONT **CreateFontIndirect** (CONST LOGFONT *lplf);

- LOGFONT 구조체가 지정하는 특성의 논리 폰트를 생성하는 함수

BOOL **ChooseFont** (LPCHOOSEFONT lpcf);

- 폰트 공용 대화상자를 여는 함수

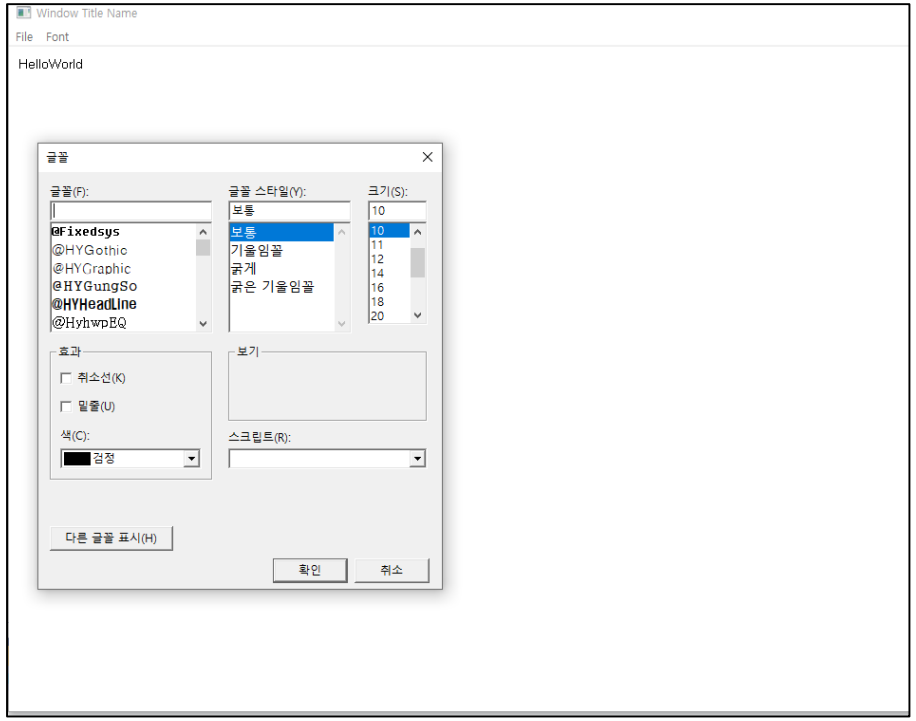
폰트 선택하기

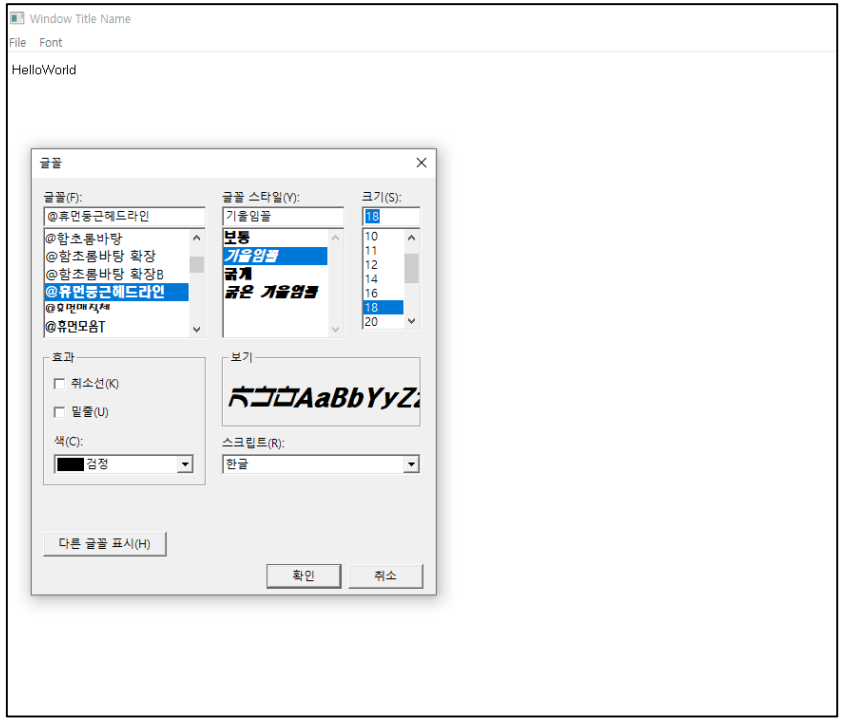
- 사용 예)

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    CHOOSEFONT          FONT;
    HFONT               hFont, OldFont;
    static LOGFONT      LogFont;
    static COLORREF     fColor;

    switch (iMsg)
    {
     case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case ID_FONTDLG:
           memset (&FONT, 0, sizeof(CHOOSEFONT));
           FONT.lStructSize = sizeof(CHOOSEFONT);
           FONT.hwndOwner = hwnd;
           FONT.lpLogFont= &LogFont;
           FONT.Flags = CF_EFFECTS | CF_SCREENFONTS;

           if (ChooseFont(&FONT)!=0) {
                fColor = FONT.rgbColors;
                InvalidateRect (hwnd, NULL, TRUE);
           }
           break;

     case WM_PAINT:
           hdc = BeginPaint(hwnd, &ps);
           hFont = CreateFontIndirect(&LogFont);
           OldFont = (HFONT) SelectObject (hdc, hFont);
           SetTextColor(hdc, fColor);
           TextOut(hdc, 10, 10, L"HelloWorld", 10);
           SelectObject(hdc, OldFont);
           DeleteObject(hFont);
           EndPaint(hwnd, &ps);
           break;
     }
}
```

폰트 선택하기

- 결과 화면


Window Title Name
File Font
HelloWorld
글꼴
글꼴(F):
글꼴 스타일(Y):
크기(S):
@Fixedsys
@HYGothic
@HYGraphic
@HYGungSo
@HYHeadLine
@HyhwpEQ
보통
기울임꼴
굵게
굵은 기울임꼴
10
11
12
14
16
18
20
효과
취소선(K)
밑줄(U)
색(C):
검정
보기
스크립트(R):
다른 글꼴 표시(H)
확인
취소


Window Title Name
File Font
HelloWorld
글꼴
글꼴(F):
글꼴 스타일(Y):
크기(S):
@휴먼둥근헤드라인
@함초롬바탕
@함초롬바탕 확장
@함초롬바탕 확장B
@휴먼모음T
기울임꼴
보통
굵게
굵은 기울임꼴
18
10
11
12
14
16
20
효과
취소선(K)
밑줄(U)
색(C):
검정
보기
AaBbYyZz
스크립트(R):
한글
다른 글꼴 표시(H)
확인
취소


Window Title Name
File Font
HelloWorld

3) 공용 대화상자: 색상 선택하기

- 색상선택하기 처리절차
  - CHOOSECOLOR 구조체 할당
  - "사용자 지정 색" 만들기
  - 색상 대화상자 띄우기-> 색상 정보 획득

```
CHOOSECOLOR COLOR;
static COLORREF tmp[16], color;

For (int i=0; i<16; i++)
      tmp[i] = 사용자 지정 색상;

memset(&COLOR, 0, sizeof(CHOOSECOLOR));
COLOR.lStructSize = sizeof(CHOOSECOLOR);
COLOR.hwndOwner = hwnd;
COLOR.lpCustColors = tmp;
COLOR.Flags = CC_FULLOPEN;

ChooseColor (&COLOR);                    //--- COLOR.rgbResult에 색상정보 저장됨
```

BOOL **ChooseColor** (LPCHOOSCOLOR color);

- 색상 공용 대화상자를 여는 함수

색상 선택하기

- CHOOSECOLOR 구조체

```
typedef struct {
        DWORD lStructSize;              //--- 구조체 크기
        HWND hwndOwner;                 //--- 메인 윈도우 핸들
        HWND hInstance;
        COLORREF rgbResult;             //--- 사용자가 대화상자에서 선택한 색상 정보
        COLORREF *lpcustColors;         //--- 색 대화상자에 사용자 지정색에 채울 색 정보 목록 (16가지)
        DWORD Flags;                    //--- 색 대화상자 초기화 하는데 사용한 플래그
        LPARAM lcustData;
        LPCCHOOKPROC lpfnHook;
        LPCTSTR lpTemplateName;
} ChOOSECOLOR, *LPCHOOSECOLOR;
```

색상 선택하기

- 사용 예)

```
CHOOSECOLOR COLOR;
static COLORREF tmp[16], color;
HBRUSH hBrush, OldBrush;
int i;

case WM_COMMAND:
        switch(LOWORD(wParam))
        {
        case ID_COLORDLG:
                for(i=0;i<16;i++)
                        tmp[i] = RGB (rand()%256, rand()%256, rand()%256);
                memset (&COLOR, 0, sizeof(CHOOSECOLOR));
                COLOR.lStructSize = sizeof(CHOOSECOLOR);
                COLOR.hwndOwner = hwnd;
                COLOR.lpCustColors = tmp;
                COLOR.Flags = CC_FULLOPEN;

                if (ChooseColor (&COLOR)!=0) {
                        color = COLOR.rgbResult;
                        InvalidateRect (hwnd, NULL, TRUE);
                }
                break;
        }
        break;
```

```
case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        hBrush = CreateSolidBrush (color);
        OldBrush = (HBRUSH)SelectObject(hdc, hBrush);
        Ellipse(hdc, 10, 10, 200, 200);
        SelectObject(hdc, OldBrush);
        DeleteObject(hBrush);
        EndPaint(hwnd, &ps);
        break;
```

결과 화면

실습 8-1

- 제목
  - 실습 2-5 (메모장 만들기)에 파일 입출력 기능 추가하기

- 내용
  - 실습 2-5 또는 2-7에서 구현한 캐럿이 있는 메모장 실습에 파일 입출력 기능을 추가한다
  - 파일 공용 대화상자를 띄워서 입/출력 할 파일을 선택하도록 하고 저장 또는 재생하도록 한다.

실습 8-2

- 제목
  - 실습 7-3 (맵툴 만들기)에 저장하기/ 읽기 기능 추가하기

- 내용
  - 실습 7-3에서 크기, 배경 이미지, 타일과 객체를 저장하도록 한다.
  - 7장 실습에 버튼 (또는 메뉴)을 추가하여 저장하기/읽기 기능을 수행하도록 한다.

오늘은

- 파일 입출력
- 공용 대화상자

- 수요일 (6월 7일)에는 7장과 8장 마지막 채점
- 다음주 월요일 (6월 12일)에는 프로젝트 발표 진행