고려대학교 ALPS 2021

C언어 스터디

Week 0x07

파일 입출력

0x01 FILE I/O

파일 입출력

오늘은 파일 입출력에 대해서 배워보도록 합시다.

앞서 배웠던 scanf, printf 함수는 표준 입출력 스트림을 이용하여 데이터를 주고받습니다.
표준 입출력을 사용하면, 프로그램을 실행하여 입력할 내용을 키보드를 이용해서 입력해야합니다.

즉, 파일을 읽어서, 파일에 내용을 쓰는 것은 불가능하겠죠?

그러므로 파일 입출력을 통해 파일의 데이터를 읽고, 파일에 데이터를 쓰는 법을 배워보도록 합시다.

0x01 FILE I/O

파일 입출력

먼저 파일 입력, 파일 출력을 위해서는,
파일을 읽던지 쓰던지 파일을 먼저 열어야 합니다.

파일을 열려면, 파일 포인터(FILE*)를 선언해야 합니다.
파일 포인터에 파일 주소를 저장한다고 생각하시면 됩니다.

그 후 fopen 함수를 이용하면 파일을 열고 파일 포인터에 주소를 부여하여 사용할 수 있습니다.
열고 파일을 모두 사용한 후에는 fclose 함수를 이용하여 파일을 닫아야 합니다.

```
FILE* fp;
fp = fopen("./input.txt","r");
// 파일 사용
fclose(fp);
```

0x01 FILE I/O

파일 입출력

fopen 함수의 첫번째 인자는 파일의 경로와 이름을 써주면 됩니다.

파일 경로를 쓸 때 ”.”은 현재 경로를 말합니다.
즉 프로그램이 실행되고 있는 위치를 말하는 것이죠.
그러므로, “./input.txt”는 프로그램이 실행되고 있는 폴더(디렉토리) 안의 input.txt 파일을 연다는 것입니다.

```
FILE* fp;
fp = fopen("./input.txt","r");
// 파일 사용
fclose(fp);
```

두번째 인자는 파일을 어떤 모드로 열 것인가에 대한 것입니다.

모드는 다음과 같습니다.
“r”: read, 파일이 존재해야 하며, 파일을 읽기만 가능
“w”: write, 파일이 있든, 없든 새로 만들어서 파일을 씁니다. 파일을 쓰기만 ㅏㄱ능
“a”: append, 파일이 없으면 만들고, 있으면 해당 파일을 열어서 뒷부분부터 씁니다.
“r+”: read+update, 파일이 존재해야 하며, 읽고 쓰기가 모두 가능
“w+”: write+update, 파일을 새로 만들며, 입 출력이 모두 가능
“a+”: append+update, 파일이 없으면 만들고, 있으면 열어서 뒷부분부터 출력, 입력은 어디서든

각 모드에 b를 넣어주면 binary file을 열 수도 있습니다(우리가 읽을수 없게 이진수로 작성된)

0x01 FILE I/O

파일 입출력

파일을 열었으면 어떻게 파일을 읽고 써야할까요 ?

파일을 읽고 쓰는 함수로
fprintf, fgets, fgetc, fputs, fputc, fscanf 등의 함수가 존재합니다.

표준 입출력에서 쓰던 익숙한 함수들에 file을 의미하는 f를 붙인 함수들입니다.

먼저 출력 시 자주 사용하는 fprintf 함수에 대해서 알아봅시다.

피보나치 수열을 파일에 출력하는 프로그램을 만들어봅시다.

0x01 FILE I/O

파일 입출력

fprintf는 파일 포인터를 인자로 넘겨주면 해당 파일에 출력을 해주는 함수입니다.
다른 fputs 등의 함수들도 비슷하게 파일 포인터를 인자로 주면 됩니다.

파일 포인터 대신에 stdout 을 인자로 넘겨주면 표준 출력으로 출력하는 것도 가능합니다.

```c
int main(){
  FILE* fp = fopen("./fibonacci.txt","w");
  int fib[100]={0,1};
  fprintf(fp, "0: %d\n1: %d\n",fib[0],fib[1]);
  for(int i=2; i<100;++i){
    fib[i]=fib[i-1]+fib[i-2];
    fprintf(fp,"%d: %d\n",i,fib[i]);
  }
  fclose(fp);
  return 0;
}
```

0x01 FILE I/O

파일 입출력

이러한 파일 입출력 함수들은 특별한 반환값을 반환하기도 합니다.
예를 들어, fscanf 함수는 데이터를 입력받는 것을 실패한다면 EOF라는 상수값을 반환합니다.
EOF는 End Of File을 의미하는 것으로 -1의 값과 동일합니다.

따라서 조건문에 fscanf을 넣어주고 EOF와 비교한다면 파일의 끝에 도달했는지 확인할 수 있습니다.
그 점을 이용하면 다음과 같이 코드를 짤 수도 있습니다.

```c
int main(){
  FILE* fp=fopen("./input.txt");
  char str[100];
  while(fscanf(fp,"%s",str)!=EOF){
    fprintf(stdout,"%s\n",str);
  }
  fclose(fp);
}
```

0x01 FILE I/O

파일 입출력

파일 입출력을 다루면서 주의할 점은 동시에 같은 파일에 접근하는 것은
예기치 못한 작동을 일으킬 수도 있다는 점입니다.

또한 아스키코드로 작성된 파일을 읽더라도 unsigned char보다는 int를 사용하는 편이 좋습니다.
왜냐하면, EOFrk -1이기 때문에 부호가 없는 자료형을 사용시 EOF를 제대로 판단하지 못할 수도
있기 때문입니다.

또한 fscanf는 공백과 탭 개행문자등을 인식하여 문자열을 자르므로 한 줄씩 읽는 것이 복잡해질 수
있기 때문에, fgets를 사용해도 됩니다.
fgets는 개행문자 \n을 읽기 전까지 계속해서 파일을 읽으므로 원하는 대로 사용할 수 있습니다.

다른 점이 있다면, fgets는 EOF대신 NULL을 사용하며,
인자 순서가 문자열, 최대 문자 수, 파일 포인터 라는 점입니다.

부족한 수업
들어 주시느라 수고 많으셨습니다!

들어주셔서 감사하고, 질문은 언제든 환영이에요

기말고사 준비로 인해 당분간 휴강하도록 하겠습니다

스터디가 9주차로 계획되어 있어, 시험 이후에
2주 간 C 언어 대신 알고리즘에 관한 내용으로
스터디를 진행할 예정입니다.

기말고사 이후에 뵙겠습니다. 다들 화이팅!