C프로그래밍및실습

생성형 AI를 이용한 메세지 전송 시스템

최종 보고서

제출일자:23.12.21.
제출자명:김건형
제출자학번:214249

1. 프로젝트 목표

1) 배경 및 필요성

운전을 하거나 급한 용무가 있을 때에는 메세지를 길게 보내는 것이 불편하다. 이런 경우에는 키워드만 간단하게 작성하여 원하는 메세지를 전달하는 기능이 필요하다 느껴본 적이 있을 것이다. 이런 기능을 구현하기 위해 생성형 AI를 이용하여 메세지 내용을 생성하고 저장해보기로 한다

2) 프로젝트 목표

생성형 AI API를 이용하여 사용자가 입력한 키워드를 가지고 메세지를 생성한다. 생성된 메세지를 배열에 저장하고 읽을 수 있도록 한다. 또한 특정 키를 입력받아 실제 메신저로 메세지를 전송하는 기능을 제공한다.

3) 차별점

기존에는 긴 메세지를 작성해야 할 때 직접 작성했다면 이 프로젝트를 통해서 메세지를 키워드만 가지고도 생성하고 전송할 수 있다. 생성형 AI의 특성상 원하지 않는 메세지가 전달될 가능성도 존재하기 때문에, 해당 메세지의 마지막에 본 메세지가 AI를 통해 생성되었다는 문구를 추가한다.

2. 기능 계획

1) 메세지 생성 기능

- 사용자의 키워드를 입력받아 메세지를 생성한다.

(1) 사용자의 키워드 입력받기

- 사용자에게 n개의 키워드를 입력받고 배열로 저장한다.

(2) 생성형 AI API 연결하기

- 생성형 AI API를 연결하여 키워드를 전송하고 생성형 AI API 기능에 포함된 말투 설정(role설정을 통해서 메세지 수신자와의 관계를 매개변수로 넘겨줄 수 있음)을 통해 사용자가 원하는 길이의 메세지를 생성한다.

#예시

수신자: 아빠

입력 키워드: 곧, 도착

생성형 AI API 매개변수: [입력 키워드를 가지고 최대 N개의 단어로 문장 생성해줘, role: 아들]

출력 메세지: 아빠 저 곧 도착해요

(3) 생성된 메세지 저장하기

- 사용자의 키워드를 통해 생성된 메세지를 배열로 저장한다.
- 저장하는 메세지의 마지막에 AI를 통해 생성된 메세지라는 문구를 추가한다.

(4) 생성된 메세지 출력, 재생성, 삭제하기

- 메세지의 번호를 선택하여 해당 메세지를 출력, 재생성 또는 삭제한다.

2) 메세지 저장 기능

- 특정 키를 함께 입력하여 프로그램을 실행하면 메세지를 파일로 저장할 수 있도록 한다.

(1) 메세지 저장 기능

- 생성한 메세지를 파일로 저장할 수 있도록 한다.

3. 진척사항

1) 기능 구현

(1) 사용자의 키워드 입력받기

- 입력: 키워드 입력 선택(번호), 키워드, 키워드 입력 종료 문자.

  출력: 사용자가 입력한 키워드

- 사용자가 키워드를 입력하고자하면 특정 번호를 입력하여 키워드를 입력할 수 있도록 하고, 키워드 입력이 종료되었음을 알리는 문자(!)를 통해서 사용자가 키워드 입력을 마치면 입력한 키워드를 출력하여 확인시켜줍니다.

- 키워드 입력 선택을 구분하는 과정에서 조건문, 키워드 입력을 받는 부분에서 반복문, 키워드와 메세지를 담고 있는 구조체, 구조체 배열을 선언하는 부분에서 동적배열로 할당하였기 때문에 포인터의 개념이 사용됨.

- 코드 스크린샷

```c
#include <stdio.h>
#include <stdlib.h>
#include "ChatGPT.h"

#define MAX_KEYWORD_SIZE 128
#define MAX_KEYWORD_COUNT 20
#define MAX_MESSEGE_SIZE 256
#define MAX_MESSEGE_COUNT 256

typedef struct KEYnMES{
  int idx;
  int keyword_num;
  char** keyword;
  char *message;
}KEYnMES;

// 키워드를 입력받고, 입력받은 키워드의 개수를 반환한다.
int input_keywords(KEYnMES keynmes) {
  // 몇 개의 키워드가 있는지 저장할 변수
  int keyword_count = 0;
  printf(
      "최대 %d 글자의 영문 키워드를 입력하세요. (입력을 마치려면 !를 입력하세요)\n",
      MAX_KEYWORD_SIZE);
  for (int i = 0; i < MAX_KEYWORD_COUNT; i++) {
    printf("%d번째 키워드: ", i + 1);
    scanf_s("%s", keynmes.keyword[i], MAX_KEYWORD_SIZE);

    if (keynmes.keyword[i][0] == '!') {
      keyword_count = i;
      printf("키워드 입력이 끝났습니다.\n\n");
      break;
    }
  }

  printf("입력하신 키워드는 총 %d개 입니다.\n", keyword_count);
  keynmes.keyword_num = keyword_count;
  for (int i = 0; i < keyword_count; i++) {
    printf("%s", keynmes.keyword[i]);
    if (i < keyword_count - 1) {
      printf(", ");
    } else {
      printf("\n");
    }
  }

  return keyword_count;
}
```

```c
int main() {
  // 메시지를 저장할 구조체를 동적할당, 초기화
  KEYnMES *keynmess =
      (KEYnMES *)malloc(MAX_MESSEGE_COUNT * (int)sizeof(KEYnMES));

  for (int i = 0; i < MAX_MESSEGE_COUNT; i++) {
    keynmess[i].idx = i + 1;
    keynmess[i].keyword =
        (char **)malloc(MAX_KEYWORD_COUNT * (int)sizeof(char *));
    for (int j = 0; j < MAX_KEYWORD_COUNT; j++) {
      keynmess[i].keyword[j] =
          (char *)malloc(MAX_KEYWORD_SIZE * (int)sizeof(char));
    }
    keynmess[i].message = (char *)malloc(MAX_MESSEGE_SIZE *
(int)sizeof(char));
  }

  // 메세지의 개수, 키워드의 개수를 저장할 변수
  int message_count = 0;
  int keyword_count = 0;

  // 선택한 메뉴를 저장할 변수
  int menu = 0;

  printf("**********자동 메세지 생성기**********\n\n");

  while (1) {
    printf("---------메뉴를 선택하세요----------\n");
    printf("1. 키워드로 메세지 생성하기\n");
    printf("2. 메세지 출력하기\n");
    printf("3. 메세지 재생성하기\n");
    printf("4. 메세지 삭제하기\n");
    printf("5. 종료하기\n");
    scanf_s("%d", &menu);
    // break;

    if (menu == 1) {
      // 키워드를 입력받고 개수를 변수에 저장한다.
      keyword_count = input_keywords(keynmess[message_count]);
      // chatgpt api를 이용해서 키워드를 통해 메세지를 작성한다.
      keynmess[message_count].message =
          chatGPT(keynmess[message_count].keyword, keyword_count);
      // 작성된 메세지를 확인하기 위해 출력해본다.
      printf("%s\n\n", keynmess[message_count].message);
      message_count++;
    }
```

(2) 생성형 AI API 연결하기

- OpenAI에서 무료로 제공하는 5$상당의 ChatGPT API를 연결함

- curl을 이용하여 https연결을 구성했음 자세한 내용은 프로젝트 repository 내 readme 참고

- API 요청했을 때 응답(답변)이 오는 것을 확인함.

(3) 문장 생성, 저장하기

- 응답을 파싱해서 KEYnMES라는 구조체 안의 message에 저장함.

- 코드 스크린샷

```c
#include <curl/curl.h>
#include <stdio.h>
#include <stdlib.h>

// 응답을 바로 출력하지 않고 저장하기 위해 선언한 함수
size_t write_callback(void *contents, size_t size, size_t nmemb,
                      char **output) {
  size_t realsize = size * nmemb;

  *output = realloc(*output, realsize + 1);
  if (*output) {
    memcpy(*output, contents, realsize);
    (*output)[realsize] = '\0';
  }

  return realsize;
}

char* chatGPT(char** keywords, int keyword_count) {

  //HTTPS 통신을 위해 선언한 CURL 구조체
  CURL *curl;
  CURLcode res;

  curl_global_init(CURL_GLOBAL_DEFAULT);
  curl = curl_easy_init();
  // curl이 정상적으로 시작 되었을 때
  if (curl) {
    curl_easy_setopt(curl, CURLOPT_URL,
                     "https://api.openai.com/v1/chat/completions");
    curl_easy_setopt(curl, CURLOPT_POST, 1L);

    // 응답을 저장하고 있을 문자열
    char *response_data = NULL;

    //api 통신을 위해 구성요소를 만드는 부분
    struct curl_slist *headers = NULL;
    headers = curl_slist_append(headers, "Content-Type: application/json");
    headers = curl_slist_append(
        headers,
        "Authorization: Bearer "
        "api-key"); // 이부분에 api key를 입력한다.
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);

    // 요청을 보낼 때 어떤 응답을 받을 것인지 결정하는 가장 중요한 부분
    char data[1000]=
        "{\"model\": \"gpt-3.5-turbo\", "
        "\"messages\": [{\"role\": \"system\",\"content\": \"you are message "
        "generator using given words\"}, "
        "{\"role\": \"user\",  \"content\": \"generate message using ";
    // 사용자가 입력한 키워드가 요청 안에 들어갈 수 있도록  strcat_s 함수 이용
    for (int i = 0; i < keyword_count; i++) {
      //printf("%d %d", strlen(data), strlen(keywords[i]));
      strcat_s(data, sizeof(data), keywords[i]);
      if (i != keyword_count - 1)
        strcat_s(data, sizeof(data), ", ");
      else
        strcat_s(data, sizeof(data), "\"");
    }
    strcat_s(data, sizeof(data),"}], \"temperature\": 0.2, \"max_tokens\":
50}");

    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data);

    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
    // 응답 데이터가 response_data 안에 저장되도록 함
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response_data);

    res = curl_easy_perform(curl);

    // 실패했을 경우의 알림 메세지
    if (res != CURLE_OK)
      fprintf(stderr, "curl_easy_perform() failed: %s\n",
              curl_easy_strerror(res));

    curl_easy_cleanup(curl);
    curl_slist_free_all(headers);

    // 응답 데이터에서 필요한 부분을 파싱하는 부분
    char *temp = strstr(response_data, "content");
    int quot_cnt = 0;
    int ret_length = 0;

    for (int i = 0; i < strlen(response_data); i++){
      if (temp[i] == '\"') {
        quot_cnt++;
      }
      if (quot_cnt == 3) {
        ret_length = i;
        break;
      }
    }

    char *ret = (char *)malloc(ret_length-10);
    for (int i = 0; i < ret_length-11; i++) {
      ret[i] = temp[i+11];
    }
    ret[ret_length-11] = '\0';

    // curl 변수를 메모리 해제시켜줌
    curl_global_cleanup();

    // 파싱된 부분을 반환함
    return ret;
  }

  curl_global_cleanup();
  return "error occured!";
}
```

(3) 문장 삭제, 재생성, 출력

- 작업하고자하는 문장의 번호를 선택하여 삭제, 재생성 또는 출력.

- 코드 스크린샷

```
else if (menu == 2) {
      printf("출력할 메세지의 번호를 선택하세요 (%d ~ %d) ", 1, message_count);
      int print = -1;
      while (1) {
        scanf_s("%d", &print);

        if (print < 1 || print > message_count) {
          continue;
        } else
          break;
      }
      printf("\n%s\n\n", keynmess[print-1].message);
    }

    else if (menu == 3) {
      printf("재생성할 메세지의 번호를 선택하세요 (%d ~ %d) ", 1, message_count
);    int remake = -1;
      while (1) {
        scanf_s("%d", &remake);

        if (remake < 1 || remake > message_count) {
          continue;
        } else
          break;
      }

      keynmess[remake-1].message =
          chatGPT(keynmess[remake-1].keyword, keyword_count);
      printf("\n%s\n\n", keynmess[remake - 1].message);
    }

    else if (menu == 4) {
      printf("삭제할 메세지의 번호를 입력하세요 (%d ~ %d) ", 1, message_count);
      int remove = -1;
      while (1) {
        scanf_s("%d", &remove);

        if (remove < 1 || remove > message_count) {
          continue;
        } else
          break;
      }
      keynmess[remove - 1].message = "삭제된 메세지입니다.";
      printf("\n삭제되었습니다!");

    }

    else if (menu == 5) {
      printf("프로그램을 종료합니다.\n");
      break;
    }
  }
}
```

(4) 메세지 저장

- 생성한 메세지들을 한 번에 파일로 저장한다.

- 파일 입출력과 관련한 내용이 사용되었다.

- 코드 스크린샷

```
else if (menu == 5) {
      printf("공백없이 저장할 파일의 이름을 입력하세요: ");
            char filename[128];
            scanf_s("%s", filename, 128);
      strcat_s(filename, sizeof(filename), ".txt");
      FILE *fp;
      fopen_s(&fp, filename, "w");
            for (int i = 0; i < message_count; i++) {
                fprintf(fp, "%d번째 메세지: %s\n", i + 1,
keynmess[i].message);
            fclose(fp);
            printf("저장되었습니다!\n");
    }
```

4. 테스트 결과

(1) 세부기능 1-1. 사용자의 키워드 입력받기

```
**********자동 메세지 생성기**********

----------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 종료하기
1
최대 128 글자의 키워드를 입력하세요. (입력을 마치려면 !를 입력하세요)
1번째 키워드: 곧
2번째 키워드: 도착
3번째 키워드: 광주역
4번째 키워드: !
키워드 입력이 끝났습니다.

입력하신 키워드는 총 3개 입니다.
곧, 도착, 광주역
```

(2) 세부기능 1-2, 1-3. 생성형 ai api 연결 및, 문장 생성, 저장하기

```
**********자동 메세지 생성기**********

---------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 종료하기
1
최대 128 글자의 키워드를 입력하세요. (입력을 마치려면 !를 입력하세요)
1번째 키워드: dad
2번째 키워드: arrive
3번째 키워드: soon
4번째 키워드: !
키워드 입력이 끝났습니다.

입력하신 키워드는 총 3개 입니다.
dad, arrive, soon
Hey Dad! Just wanted to let you know that I'll be arriving soon. Can't wait to see you!
```

(3) 세부기능 1-4. 생성된 문장, 출력, 재생성, 삭제하기

```
---------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 종료하기
2
출력할 메세지의 번호를 선택하세요 (1 ~ 1) 1

Hey Dad! Just wanted to let you know that I'll be arriving soon. Can't wait to see you!
```

```
---------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 종료하기
3
재생성할 메세지의 번호를 선택하세요 (1 ~ 1) 1

Hey Dad! Just wanted to let you know that I'll be arriving soon. Can't wait to see you!
```

```
---------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 종료하기
4
삭제할 메세지의 번호를 입력하세요 (1 ~ 1) 1

삭제되었습니다!---------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 종료하기
2
출력할 메세지의 번호를 선택하세요 (1 ~ 1) 1

삭제된 메세지입니다.
```

(4) 세부기능 2-1. 생성된 문장 저장하기

```
---------메뉴를 선택하세요----------
1. 키워드로 메세지 생성하기
2. 메세지 출력하기
3. 메세지 재생성하기
4. 메세지 삭제하기
5. 생성된 메세지 파일로 저장하기
6. 종료하기
5
공백없이 저장할 파일의 이름을 입력하세요: kimchi_mom
저장되었습니다!
```

5. 계획 대비 변경 사항

- api 통신에 있어서 텍스트 인코딩이 utf-8로 되어야하지만 visual studio의 한글 기본 인코딩은 euc-kr이다. 때문에 한국어로 요청을 보낼 수 없었다. iconv 패키지로 해결할 수 있었으나 프로젝트를 실행하기 위한 난이도가 너무 상승하여 제외할 수 밖에 없었다.

- 메세지를 송신하는 기능을 추가하려고 했으나, 위와 마찬가지로 프로젝트를 실행하기 위한 난이도가 너무 상승하기 때문에 메세지를 저장하는 기능으로 변경, 구현하였다.

6. 느낀점

이번 프로젝트에서 C를 통해 기초부터 기능을 구현하며 오류가 없는 프로그램을 작성하려면 얼마나 많은 것을 신경써야 하는지를 다시 한 번 느끼게 되었다. 예를 들어 인공지능에서 많이 사용하는 파이썬이나 알고리즘 공부를 하며 사용했던 C++에서는 문자열을 합치며 문자열이 들어가는 공간의 크기를 함수에 같이 넣어줘야 한다는 것을 생각해본 적이 없지만, 순수 C에서는 문자열이 들어가는 공간을 같이 strcat_s 함수에 넣어주면서 구현이 훨씬 어렵지만 더욱 정확한 코딩이라는 생각이 들었다.

이렇게 기능 구현에서 복잡하고 신경써야할 것이 많은 C임에도 생성형 AI를 이용해서 프로젝트를 진행해보고 싶었던 이유는 복잡한 API를 다루면서 오히려 저수준에서 API와 요청, 응답에 대해서 더 자세하게 이해할 수 있을 것이라 생각했기 때문이고, 실제로 프로젝트를 진행하며 기대했던 부분에 대해 학습할 수 있어서 만족스러웠다. 물론 구현에 있어서 문자열을 다루는 부분이 많았기 때문에 동적할당을 통해서 문자열 복사, 병합 등의 처리가 복잡했지만, 결과적으로 정상적으로 작동하는 프로그램을 만들 수 있었음에 다행이라고 생각한다.

아쉬웠던 점은 ChatGPT API는 원래 한글로도 요청을 보낼 수 있는데, visual studio의 한글 인코딩 기본 설정이 cp949(혹은 euc-kr)로 되어 있기 때문에 utf-8로만 요청을 받는 ChatGPT API로 정상적인 요청을 보낼 수 없었기 때문에 다른 라이브러리를 사용하며 실행 난이도를 높이기보다는 영문으로 요청을 보내는 것으로 변경했던 점과 같은 이유로 메세지 전송 기능을 메세지 저장기능으로 변경했던 점이 있었다.

이후에 순수 C로 ChatGPT에 관한 프로젝트를 이어서 하게 된다면 상기된 문제들을 해결해서 시작하며 기획했던 모든 기능을 갖춘 프로그램을 만들어보고싶다.