C 언어 치트시트

기본 구조 및 컴파일

C 프로그램은 필요한 헤더 파일 포함, 프로그램의 시작점인 `main` 함수, 그리고 다양한 실행 구문으로 구성됩니다.

```c
#include <stdio.h> // 표준 입출력 라이브러리 포함

// 프로그램 실행이 시작되는 메인 함수
int main() {
    printf("Hello, World!\n"); // 화면에 문자열 출력
    return 0; // 프로그램 정상 종료 반환
}
```

- 컴파일 및 실행 (GCC 기준):

```bash
gcc -o hello hello.c # 컴파일하여 실행 파일 생성
./hello              # 생성된 실행 파일 실행
```

변수와 자료형

- 기본 데이터 타입:
  - `int`: 정수형 (정수 저장)
  - `float`, `double`: 실수형 (부동 소수점 숫자 저장)
  - `char`: 문자형 (단일 문자 저장)
  - `void`: 타입이 없음을 나타냄
- 자료형 한정자: `short`, `long`, `signed`, `unsigned`
- 변수 선언 방식: `자료형 변수명;` (예: `int age;`)
- 상수 정의: `const double PI = 3.14;`
- 배열 선언: `int numbers[10];` (정수 10개를 저장하는 배열)
- 문자열 처리: `char greeting[] = "Hello";` (마지막이 널 문자 `\0`로 끝나는 문자 배열)

주요 연산자

- 산술 연산자: `+`, `-`, `*`, `/`, `%` (나머지), `++` (증가), `--` (감소)
- 관계 연산자: `==` (같음), `!=` (다름), `>`, `<`, `>=`, `<=`
- 논리 연산자: `&&` (논리곱, AND), `||` (논리합, OR), `!` (부정, NOT)
- 비트 연산자: `&` (AND), `|` (OR), `^` (XOR), `~` (NOT), `<<`, `>>` (시프트)
- 할당 연산자: `=`, `+=`, `-=`, `*=`, `/=`, `%=`
- 기타 연산자: `sizeof` (크기 확인), `&` (주소 추출), `*` (포인터 역참조), `?:` (조건 삼항 연산자)

제어 흐름 (조건문 및 반복문)

- `if-else` 조건문:

```c
if (조건식) {
    // 조건이 참일 때 실행
} else if (다른조건) {
    // 다른 조건이 참일 때 실행
} else {
    // 모든 조건이 거짓일 때 실행
}
```

- `switch` 선택문:

```c
switch (변수) {
    case 값1:
        // 값1일 때 실행
        break;
    case 값2:
        // 값2일 때 실행
        break;
    default:
        // 일치하는 값이 없을 때 실행
}
```

- 반복문:
  - `for` 루프: `for (초기화; 조건식; 증감식) { ... }`
  - `while` 루프: `while (조건식) { ... }`
  - `do-while` 루프: `do { ... } while (조건식);` (최소 1회 실행 보장)
- 흐름 제어 키워드: `break` (루프 탈출), `continue` (다음 반복 진행)

함수 정의 및 호출

- 함수 선언 (원형): `반환타입 함수명(매개변수타입 매개변수1, ...);`
- 함수 정의 예시:

```c
int add(int a, int b) {
    // 함수 내부 로직
    return a + b; // 결과 값 반환
}
```

포인터 (Pointers)

포인터는 메모리 주소를 직접 저장하는 특수한 변수입니다.

- 포인터 선언: `자료형 *포인터명;` (예: `int *ptr;`)
- 주소 연산자 (`&`): 일반 변수의 메모리 주소를 가져옵니다. `ptr = &age;`
- 역참조 연산자 (`*`): 포인터가 가리키는 주소에 저장된 실제 값을 가져옵니다. `val = *ptr;`

구조체 (Structures)

서로 다른 타입의 데이터를 하나의 논리적 단위로 묶어 관리합니다.

```c
struct Person {
    char name[50];
    int age;
};

// 구조체 변수 사용 예시
struct Person p1;
strcpy(p1.name, "Alice");
p1.age = 25;
```

- `typedef` 활용: 복잡한 타입 이름을 간결하게 재정의합니다. `typedef struct Person Person;`

전처리기 지시자 (Preprocessor)

컴파일이 시작되기 전에 소스 코드를 가공하는 역할을 합니다.

- `#include <파일명>`: 시스템 표준 헤더 파일 포함
- `#include "파일명"`: 사용자 정의 헤더 파일 포함
- `#define 이름 값`: 매크로 상수 또는 매크로 함수 정의
- 조건부 컴파일: `#ifdef`, `#ifndef`, `#if`, `#else`, `#elif`, `#endif`

동적 메모리 관리

실행 중에 필요한 만큼 메모리를 할당하고 해제합니다. (`<stdlib.h>` 포함 필요)

- `malloc(크기)`: 지정한 바이트만큼 메모리를 할당합니다.
- `calloc(개수, 크기)`: 메모리 할당 후 모든 비트를 0으로 초기화합니다.
- `realloc(포인터, 새크기)`: 이미 할당된 메모리 영역의 크기를 조정합니다.
- `free(포인터)`: 사용이 끝난 메모리 영역을 해제하여 반환합니다.

파일 입출력 (File I/O)

파일 시스템의 데이터를 읽거나 쓸 때 사용합니다. (`<stdio.h>` 포함 필요)

- `FILE *fp;`: 파일을 제어하기 위한 파일 포인터 선언
- `fopen("파일명", "모드")`: 파일 열기 (모드: `r` 읽기, `w` 쓰기, `a` 추가)
- `fclose(fp)`: 열려 있는 파일 닫기
- `fprintf(fp, ...)` / `fscanf(fp, ...)`: 파일 형식화 입출력
- `fgetc(fp)` / `fputc(c, fp)`: 단일 문자 단위 입출력
- `fgets(str, n, fp)` / `fputs(str, fp)`: 문자열 단위 입출력