Programming Language

컴퓨터개론

(Introduction to Computer Systems)

GEN1030

C Language

C 언어

- 1972년 데니스 리치(Dennis Ritchie)가 설계
  - 유닉스(Unix) OS 개발 위한 언어
  - 현재 사용되는 대부분의 운영체제, 컴파일러 등이 C 언어로 구현 됨
  - 자원이 제한된 환경에서 사용되는 프로그램은 여전히 C 언어를 사용해서 많이 구현 됨(ex. 임베디드, IoT)
  - 현대 많은 프로그래밍 언어의 문법적 모태

- 특징
  - 컴파일 과정 필요
  - 정적 타이핑(Static Typing)
    - 데이터의 타입을 명확히 선언해야 함
  - 하드웨어와 메모리 직접 제어가 가능

Python vs. C 언어

- Python
  - 문법이 (비교적) 간단
  - 동적 타이핑(Dynamic Typing)
  - 인터프리터 언어
  - 메모리 자동 관리

- C 언어
  - 정적 타이핑(Static Typing)
  - 컴파일 언어
  - 메모리 직접 접근 및 사용 가능
  - 성능이 좋음

    같은 작업 시 더 빨리 끝남
    불필요한 작업이 적음(ex. 타입 검사)

C – 프로그램 작성 및 실행

- 소스 코드(Source code) 작성

- 컴파일(Compile)
  - 목적 파일(Object file) 생성 – 아직 실행은 불가능
- 링크(Link)
  - 여러 목적 파일 및 라이브러리 코드 합침
  - 실행 파일(executable) 생성

→ **gcc (GNU Compiler Collection), Clang 등 사용**

- 실행

C – 프로그램 작성 및 실행

- gcc
  - Ex) Source file: main.c
  - 컴파일(Compile)
    - gcc -c main.c → main.o (Object file)
  - 링크(Link)
    - gcc main.o -o test_program → (Executable)
  - 컴파일 + 링크
    - gcc main.c -o test_program → (Executable)
    - gcc -Wall main.c -o test_program

      *Wall(Warning all) option: Warning 메세지 출력
- 실행
  - ./test_program

C – 프로그램 작성 및 실행

- gcc
  - Ex) Source file: main.c func.c (2개)
  - 컴파일(Compile)
    - gcc -c main.c → main.o (Object file)
    - gcc -c func.c → func.o
  - 링크(Link)
    - gcc main.o func.o -o test_program → (Executable)
  - 컴파일 + 링크
    - gcc main.c func.c -o test_program → (Executable)
- 실행
  - ./test_program

C – 기본 문법

- 문장 끝에는 ; 붙임
  - 세미콜론(Semicolon)
- 코드 블록은 { } 로 묶음
  - 중괄호(Curly bracket)
- 대소문자 구분
- 주석(Comment)은 // 혹은 /**/ 사용
  - // 한 줄 주석
  - /* 여러 줄
    주석 */

C – 기본 문법

- 변수(Variable)
  - 데이터 저장 위한 메모리 공간
  - 자료형을 명시적으로 선언
  - 이름 붙여서 사용

- 선언 방법

자료형 변수이름;

int a;

자료형 변수이름 = 초기값;

int a = 20;

vs. Python

a = 20

(Dynamic typing)

C – 기본 문법

- 자료형(Data Type)
  - 자료형 크기가 시스템에 따라 달라질 수 있음

- 정수형(소수점이 없는 숫자)
  - **int**
    - 보통 4 byte
    - 범위: -2,147,483,648 ~ 2,147,483,647
    - unsigned int? 음수 없음
      - 범위: 0 ~ 4,294,967,295

```
int a = 20;
```

  - **long**
    - 4 byte (ex. Windows) 혹은 8 byte (ex. Linux)

```
long b = 20;
```

C – 기본 문법

- 자료형(Data Type)
  - 자료형 크기가 시스템에 따라 달라질 수 있음

- 실수형(소수점이 있는 숫자)
  - **float**
    - 보통 4 byte

```
float x = 3.523951;
```

  - **double**
    - 보통 8 byte
    - 정밀도가 높음

```
double y = 11.2520478;
```

C – 기본 문법

- 자료형(Data Type)
  - 자료형 크기가 시스템에 따라 달라질 수 있음

- 문자형
  - **char**
    - 문자 하나를 저장
    - 작은 따옴표 사용(' ')
    - 보통 1 byte

```
char alphabet = 'A';
```

C – 기본 문법

- 자료형(Data Type)
  - 자료형 크기가 시스템에 따라 달라질 수 있음

- 배열(Array)
  - 같은 자료형의 값을 여러 개 저장
  - 배열 크기 대괄호 [] 사용해서 지정
  - 인덱스(index)로 데이터 접근

```
int a[5] = {0, 1, 2, 3, 4};

int a[] = {0, 1, 2, 3, 4};
```

```
a[2]   → 2
a[2] = 6;
a[2]   → 6
```

C – 기본 문법

- 자료형(Data Type)
  - 자료형 크기가 시스템에 따라 달라질 수 있음

- 배열(Array)
  - 같은 자료형의 값을 여러 개 저장
  - 배열 크기 대괄호 [] 사용해서 지정
  - 인덱스(index)로 데이터 접근

```
char b[3] = {'x', 'y', 'z'};

char b[] = {'x', 'y', 'z'};
```

```
char b[] = "xyz";
```

문자열(String)

- 문자열 전용 자료형 없음
- 큰따옴표 사용 (" ")

메모리: ['x']['y']['z']['\0']

\0 : 문자열의 끝을 나타내는 문자 (null character)

변수 선언 – Python vs. C

Python

```
x = 10
y = 3.14
name = "Alice"
```

Dynamic typing

C

```
int x = 10;
float y = 3.14;
char name[] = "Alice";
```

Static typing

C – 기본 문법

- 입력
  - scanf() 함수

```
int x;
scanf("%d", &x);
```

%d: 정수를 입력 받음
&x: x의 주소를 전달
&: Address operator

- 출력
  - printf() 함수

```
printf("Hello");
printf("%d", 10);
printf("%f", 3.14);
printf("%c", 'A');
```

형식지정자(format specifier)
- %d: 정수
- %f: 실수
- %c: 문자
- %s: 문자열

C – 기본 문법

- 조건문(Conditional Statement)
  - 주어진 조건(Condition)이 참(True)인지 거짓(False)인지에 따라 프로그램 실행 흐름을 바꾸는 문법
  - 조건은 괄호 ( ) 안에 작성
  - 실행할 코드는 중괄호 { } 안에 작성

```
1   int x = 10;
2
3   if (x > 0) {
4       printf("positive\n");      // \n: 줄바꿈(newline) 문자
5   } else if (x == 0) {
6       printf("zero\n");
7   } else {
8       printf("negative\n");
9   }
```

C – 기본 문법

- 반복문(Loop Statement)
  - for, while

```
for (int i = 0; i < 3; i++) {
    printf("%d\n", i);
}
```

초기식: int i = 0 i를 0으로 시작
조건식: i < 3 i < 3 동안 반복
증감식: i++ 반복할 때 마다 i가 1씩 증가

```
int i = 0;
while (i < 3) {
    printf("%d\n", i);
    i++;
}
```

조건식: i < 3
조건이 참인 동안 계속 반복

C – 기본 문법

- 함수(Function)
  - 특정 작업 수행하는 코드 묶음
  - 반환타입 및 매개변수 타입 명시

```
반환타입  함수이름(매개변수타입 매개변수, ...) {
    처리
    return 반환값;
}
```

```
int add(int a, int b) {
    int result = a + b;
    return result;
}
```

cf) Python

```
def add(a, b):
    result = a + b
    return result
```

C – 기본 문법

- 함수(Function)
  - 호출(call) 되어야 실제 실행 됨

```
int add(int a, int b) {
    int result = a + b;
    return result;
}

int main() {
    int x = add(3, 5);
    return 0;
}
```

main() 함수

- C 프로그램 시작점(Entry point)
- return 0: 정상 종료되었음을 알림

C – 기본 문법

- 라이브러리(Library)
  - 특정 기능을 수행하기 위해 미리 작성 된 코드 묶음/집합체
  - 복잡한 기능을 직접 구현하지 않아도 됨

  - #include 사용

    #include <stdio.h>
      표준 입출력 라이브러리(ex. scanf, printf)

    #include <stdlib.h>
      표준 라이브러리 - 메모리 관리(ex. malloc), 자료형 변환(ex. atoi) 등

    #include <string.h>
      문자열 처리 - 문자열 조작/복사/비교 등 (ex. strcpy, strcmp)

C – Examples

- 두 수의 합
  - 입출력

```
1    #include <stdio.h>
2
3    int main() {
4        int a, b;
5        scanf("%d %d", &a, &b);
6        printf("%d₩n", a + b);
7        return 0;
8    }
```

C – Examples

- 짝수 / 홀수 판별
  - 입출력
  - 조건문

```
1   #include <stdio.h>
2
3   int main() {
4       int x;
5
6       scanf("%d", &x);
7
8       if (x > 0) {
9           printf("Positive number₩n");
10      } else if (x == 0) {
11          printf("Zero₩n");
12      } else {
13          printf("Negative number₩n");
14      }
15      return 0;
16  }
```

C – Examples

- 반복문

```c
#include <stdio.h>

int main() {
    int i;

    for (i = 1; i <= 5; i++) {
        printf("%d₩n", i);
    }

    return 0;
}
```

```c
#include <stdio.h>

int main() {
    int scores[5] = {5, 3, 1, 10, 6};
    int i;

    for (i = 0; i < 5; i++) {
        printf("%d₩n", scores[i]);
    }

    return 0;
}
```

C – 기본 문법

- 함수

```
#include <stdio.h>

int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

int main() {
    int a = 10;
    int b = 20;

    printf("Max = %d₩n", max(a, b));

    return 0;
}
```

Summary

- C language
  - Data type
  - Input/Output
  - Conditional / Loop statement
  - Function / library
  - Eamples