Lecture 9. 포인터 (1)

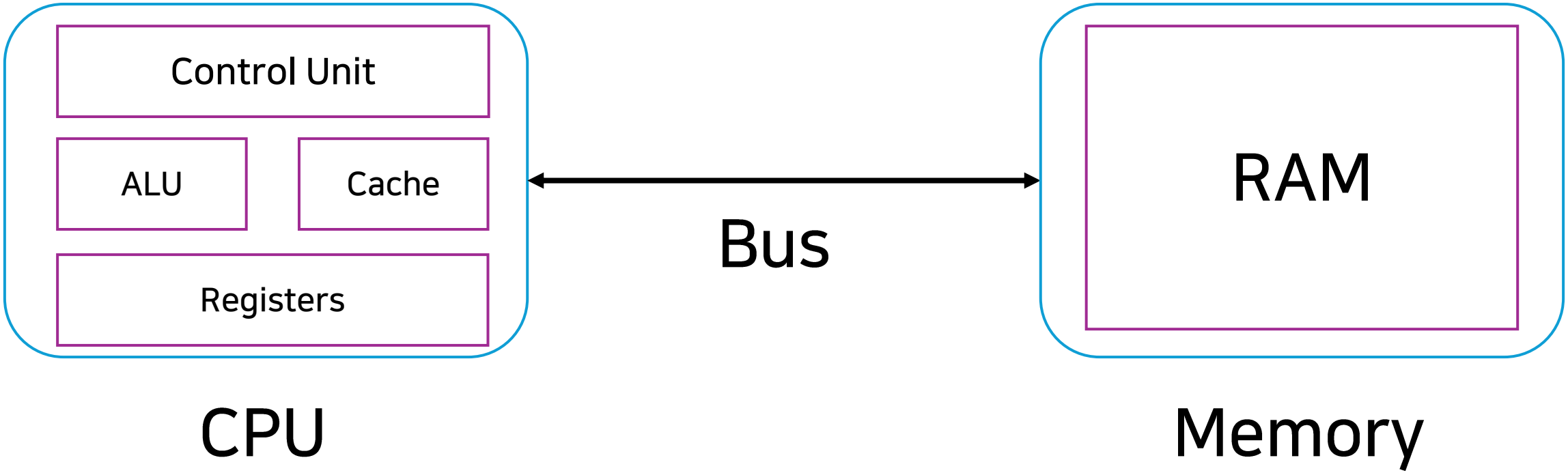
BAE SEKANG(배세강)

Contents

Pointer

♣ Computer System & Memory

- 메모리와 CPU가 정보를 주고 받는 형태
- 메모리는 주소(address)를 가지고 있음



Pointer

♣ 포인터

- 주소를 저장하는 변수
- 변수 앞에 *를 붙여서 선언

```
#include <stdio.h>

int main() {
    int a = 10;
    int *p = &a;

    return 0;
}
```

Example)

Address	Variable	Value
0x1000	a	10
0x2000	p	0x1000

Pointer

♣ Reference Operator

- 주소 연산자: &

```
#include <stdio.h>

int main() {
    int a = 10;
    int *p = &a; // 변수 a가 저장된 메모리 주소

    printf("%p₩n", p); // a의 주소 출력; %p: 메모리 주소를 16진수 형태로 출력
    return 0;
}
```

- scanf(“%d”, &a);는 입력한 값을 a의 주소에 해당하는 공간에 저장한다.

Pointer

♣ Reference Operator

- 포인터 변수도 메모리에 저장된다. (p의 주소가 존재)

```
#include <stdio.h>

int main() {
    int a = 10;
    int *p = &a;
    printf("address of a: %p₩n", p); // a의 주소 출력
    printf("address of p: %p₩n", &p); // p의 주소 출력

    return 0;
}
```

Pointer

♣ Dereference Operator

- 역참조 연산자: *
- 포인터가 가리키는 주소에 위치한 값을 읽는다.

```
#include <stdio.h>

int main() {
    int a = 10;
    int *p = &a;
    printf("%d₩n", *p);

    return 0;
}
```

p : a의 주소
*p : a의 값

Pointer

♣ Dereference Operator

- 역참조를 하여 값을 수정할 수도 있다.

```
#include <stdio.h>

int main() {
    int a = 10;
    int *p = &a;
    *p = 30;
    printf("%d₩n", a); // 30 출력

    return 0;
}
```

Pointer

♣ Pointer 활용 코드

- 예시: 문자열의 소문자 알파벳을 대문자로 변경하는 코드

```
#include <stdio.h>

int main() {
    char str[] = "hello world"; // 컴파일러가 맨 뒤에 ₩0를 붙여준다.
    char *p = str;

    while (*p != '₩0') {
        if (*p >= 'a' && *p <= 'z') {
            *p = *p - 32;
        }
        p++;
    }

    printf("%s", str);
    return 0;
}
```

Pointer

[Quiz] – 다음 프로그램의 출력 결과는?

```
#include <stdio.h>

int main() {
    int x = 100;
    int *p = &x;

    *p -= 50;
    x = *p * 2;
    p += 10;

    printf("%d", x);

    return 0;
}
```

Pointer

[Quiz] – Answer: 100

```
#include <stdio.h>

int main() {
    int x = 100;
    int *p = &x; // 포인터 p가 x의 주소를 가리킨다

    *p -= 50; // x -= 50과 동일하다
    x = *p * 2; // x *= 2; x = 100
    p += 10; // x값에 영향을 주지 않는다

    printf("%d", x); // 100

    return 0;
}
```

Pointer and Array

♣ Array Name

- 배열의 이름은 배열의 시작 주소와 같다.
- arr은 &arr[0]과 같다.

```
#include <stdio.h>

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int *p = arr;

    printf("%d₩n", *p);
    printf("%d₩n", arr[0]);

    return 0;
}
```

Pointer and Array

♣ Pointer Arithmetic

- 포인터 연산의 결과는 자료형 크기에 따라 결정된다.

```
#include <stdio.h>

int main() {
    char c = 'A';
    int i = 10;
    double d = 3.14;

    char *pChar = &c;
    int *pInt = &i;
    double *pDouble = &d;

    printf("--- 자료형별 +1 연산 결과 ---₩n");
    printf("char   포인터: %p -> %p₩n", pChar, pChar + 1);              // char: 1B
    printf("int    포인터: %p -> %p₩n", pInt, pInt + 1);                // int: 4B
    printf("double 포인터: %p -> %p₩n", pDouble, pDouble + 1);    // double: 8B

    return 0;
}
```

출력 예시)

```
--- 자료형별 +1 연산 결과 ---
char   포인터: 0x7ffef4795783 -> 0x7ffef4795784
int    포인터: 0x7ffef4795784 -> 0x7ffef4795788
double 포인터: 0x7ffef4795788 -> 0x7ffef4795790
```

Pointer and Array

♣ Pointer Arithmetic

- 포인터 연산을 통해 배열에 접근할 수 있다.

```
#include <stdio.h>

int main() {
    int arr[3] = {10, 20, 30};
    int *p = arr;

    for (int i = 0; i < 3; i++) {
        printf("%d ", *(p + i));
    }
    printf("₩n");

    *(p + 1) = 50;
    printf("%d", arr[1]);

    return 0;
}
```

Pointer and Array

♣ String Literal

- 문자열 상수는 메모리에 저장된 첫 번째 문자의 주소값과 같다.
- string Literal은 Read-Only 영역에 저장된다.

```
#include <stdio.h>

int main() {
    char *p = "Hello";

    printf("%s₩n", p);
    printf("%s₩n", p + 1);
    printf("%c₩n", *p);
    printf("%c₩n", *(p + 3));

    return 0;
}
```

```
Hello
ello
H
l
```

Pointer and Array

♣ string.h

- strcpy(*dest, *src): src의 문자열을 dest로 옮긴다.
- 복사는 ₩0을 만날 때까지 진행한다.

```
#include <stdio.h>
#include <string.h>

int main() {
    char origin[] = "C Programming";
    char copy[20];
    char *ptr = origin;

    strcpy(copy, ptr);
    printf("%s", copy);

    return 0;
}
```

Pointer and Array

♣ string.h

- strcmp(*s1, *s2): s1과 s2가 가리키는 문자열을 ASCII 값으로 하나씩 비교하며 이동한다.
- 0: 두 문자열 일치, 양수: s1이 사전 순으로 뒤, 음수: s1이 사전 순으로 앞

```
#include <stdio.h>
#include <string.h>

int main() {
   char *s1 = "abc";
   char *s2 = "abc";

   if (strcmp(s1, s2) == 0) {
      printf("Equal");
   }

   return 0;
}
```

Pointer and Array

[Quiz] – 다음 프로그램의 출력 결과는?

```
#include <stdio.h>
#include <string.h>

int main() {
    char a[] = "Hello, World!";
    char b[] = "Hello, World!";

    printf("%d₩n", (a == b) + strcmp(a, b));
    return 0;
}
```

Pointer and Array

[Quiz] – Answer: 0

```
#include <stdio.h>
#include <string.h>

int main() {
    char a[] = "Hello, World!";
    char b[] = "Hello, World!";

    // a과 b는 내용은 같지만 서로 다른 메모리 공간에 저장된 배열이므로, 주소 값이 다르다
    // strcmp 함수는 문자열 내용이 같으면 0을 반환한다
    printf("%d₩n", (a == b) + strcmp(a, b));
    return 0;
}
```

Pointer and Array

[Practice]

문자열 A, B를 입력할 때, 사전 순으로 더 빠른 단어를 출력한다. 만약 A와 B가 같다면 “Equal”을 출력한다. (A, B 모두 100글자 이하이다)

[입력 1]
array pointer

[입력 2]
string string

[출력 1]
array

[출력 2]
Equal

Pointer and Array

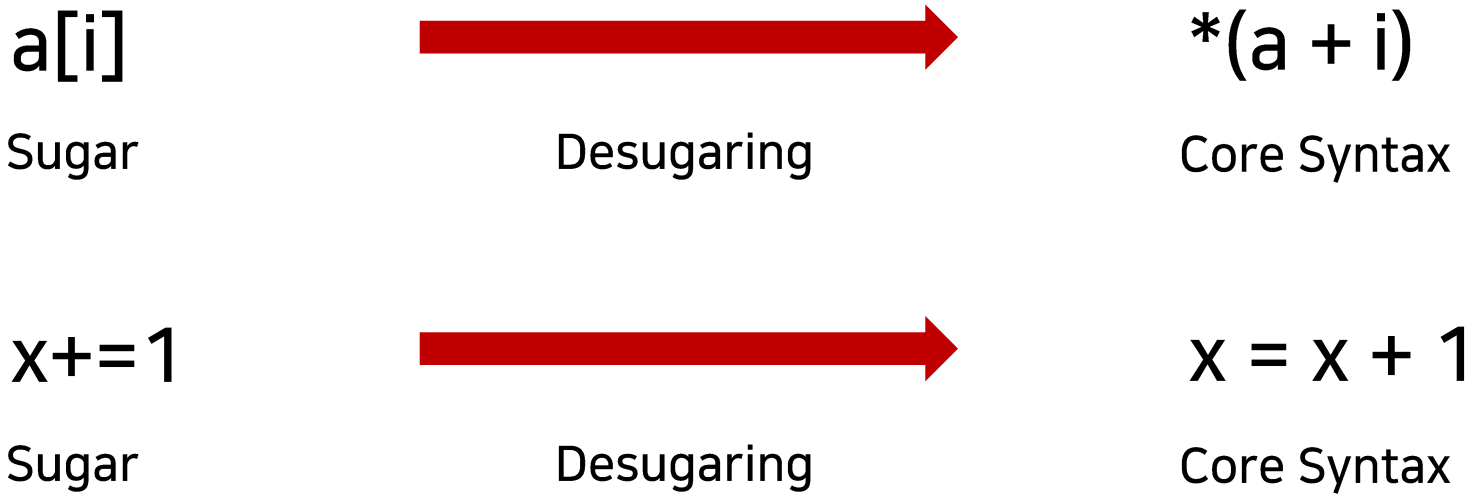
[Practice] - Answer

```
#include <stdio.h>
#include <string.h>

int main() {
    char a[101], b[101];

    scanf("%s %s", a, b);

    int result = strcmp(a, b);

    if (result < 0) {
        printf("%s₩n", a);
    } else if (result > 0) {
        printf("%s₩n", b);
    } else {
        printf("Equal₩n");
    }

    return 0;
}
```

Pointer and Array

♣ Syntactic Sugar

- 언어의 기능을 확장하지는 않으면서 프로그래머가 코드를 더 효율적이고 가독성 있게 작성할 수 있도록 설계된 문법
- Desugaring: Syntactic Sugar를 걷어내어 Core Syntax로 변환하는 과정
- C언어의 배열 문법이 대표적인 Syntactic Sugar



Pointer and Array

♣ Syntactic Sugar

- 2[p]는 *(2 + p)로 변환된다.

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int *p = &arr[1];

    printf("%d", 2[p]);

    return 0;
}
```

Parameter Passing

♣ Call by Value

- 변수의 복사본을 전달하는 방식; 함수 내부에서 값이 바뀌어도 원본은 그대로

```
#include <stdio.h>

void func(int n) {
    n = 100;
}

int main() {
    int a = 10;
    func(a);
    printf("%d", a); // 10
    return 0;
}
```

Parameter Passing

♣ Call by Reference

- 변수의 주소를 전달하는 방식; 함수 내부에서 원본을 수정할 수 있다
- C언어는 Call by Value방식으로 구현되어 있다 (포인터로 Call by Reference 효과를 낼 수 있을 뿐)
- C언어의 함수에 전달되는 모든 parameter는 그 변수 값의 복사본이다

```
#include <stdio.h>

void func(int *p) {
    *p = 100;
}

int main() {
    int a = 10;
    func(&a);
    printf("%d", a); // 100
    return 0;
}
```

```
#include <stdio.h>

void f(int *p) {
    p = NULL; // NULL pointer
}

int main(void) {
    int a = 3;
    int *q = &a;
    f(q);
    printf("%p₩n", q); // 여전히 &a
}
```

<Call by Reference 예시>

- C++; void f(int& x)
- C#; void f(ref int x)
- PHP; function f(&$x)
- Pascal; procedure f(var x: integer)
- …

Parameter Passing

♣ Pointer Parameter

- 함수의 parameter로 주소값을 받는 경우

```
#include <stdio.h>

void change(int *p) {
   *p = 10;
   *p += 5;
}

int main(void) {
   int a = 3;

   change(&a);

   printf("%d₩n", a); // 15
}
```

Parameter Passing

♣ Pointer Parameter

- 문자열과 배열도 parameter로 받을 수 있다

```
#include <stdio.h>

void printArray(int *ptr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", ptr[i]);
    }
}

int main() {
    int data[] = {1, 2, 3};
    printArray(data, 3); // data는 &data[0]를 전달한다
    return 0;
}
```

Parameter Passing

♣ Swap Function

- a와 b의 값을 서로 바꿀 수 있는 함수

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    swap(&a, &b);
    printf("%d %d₩n", a, b);

    return 0;
}
```

Parameter Passing

♣ Swap Function

- xor을 3번 연산해도 swap이 가능하다.
  (단, a와 b가 서로 다른 메모리 위치를 가리킬 때만 가능)

```
#include <stdio.h>

void swap(int *a, int *b) {
   *a ^= *b;  // STEP 1
   *b ^= *a;  // STEP 2
   *a ^= *b;  // STEP 3
}

int main() {
   int a, b;
   scanf("%d %d", &a, &b);
   swap(&a, &b);
   printf("%d %d₩n", a, b);
   return 0;
}
```

[Properties of XOR operation]

1. $a \oplus a = 0$
2. $a \oplus 0 = a$
3. $a \oplus b = b \oplus a$

pf) $a = A, b = B;$

$a = A \oplus B;$ // STEP 1

$b = B \oplus (A \oplus B) = A;$ // STEP 2

$a = (A \oplus B) \oplus A = B;$ // STEP 3

Parameter Passing

[Practice]

문자열을 대문자로 변환하는 함수 toUpper(char *c)를 구현하고, 이를 활용하여 사용자가 입력한 단어 S의 소문자 알파벳을 대문자로 변환하여 출력한다. (단 S는 100글자 이내이다.)

[입력 1]
hello

[입력 2]
Awesome

[출력 1]
HELLO

[출력 2]
AWESOME

Parameter Passing

[Practice] - Answer

```
#include <stdio.h>

void toUpper(char *c) {
   while (*c != '₩0') {
      if (*c >= 'a' && *c <= 'z') {
         *c = *c - 32;
      }
      c++;
   }
}

int main() {
   char s[101];

   scanf("%s", s);

   toUpper(s);
   printf("%s₩n", s);

   return 0;
}
```

Exercise

[Problem 1]

단어 S, A, B를 순서대로 입력 받는다. S 내에 있는 모든 A를 B로 바꾸는 함수 replaceChar를 구현하고, 이를 활용해 치환 과정이 이루어진 S를 출력한다. (단 $0 \leq |A|, |B| \leq |S| \leq 100$이다.)

[입력 1]
apple a z

[입력 2]
ABCDBBC BC DF

[출력 1]
zpple

[출력 2]
ADFDBDF

Exercise

[Problem 1] - Answer

```
#include <stdio.h>

void replaceChar(char *s, char *oldW, char *newW) {
   int len = 0;
   while (oldW[len] != '₩0') len++;

   while (*s != '₩0') {
      char *p1 = s, *p2 = oldW;

      while (*p2 != '₩0' && *p1 == *p2) {
         p1++;
         p2++;
      }
      if (*p2 == '₩0') {
         for (int i = 0; i < len; i++) s[i] = newW[i];
         s += len;
      } else s++;
   }
}

int main() {
   char s[101], a[101], b[101];
   scanf("%s %s %s", s, a, b);
   replaceChar(s, a, b);
   printf("%s₩n", s);

   return 0;
}
```

THANK YOU!

Any Question?