고급C프로그래밍

04 고급 포인터(1/2)

메모리 할당 함수
- 포인터 리뷰 (1/3)

- **포인터 기본 내용**

❶ 컴퓨터의 모든 메모리에는 주소(Address)가 지정

❷ int aa[3];과 같이 배열을 선언

- 배열 aa는 변수가 아닌 메모리의 주솟값 그 자체를 의미, '포인터 상수'

❸ 포인터 변수란 "주소를 담는 그릇(변수)"

- 포인터 변수 선언 : int *p; 또는 char *p;와 같이 '*'를 붙여서 선언

❹ 포인터 변수에는 주소만 대입

- 주소의 표현 : 변수 앞에 '&' 붙이기

메모리 할당 함수
– 포인터 리뷰 (2/3)

- **예제 12-1 (1/2)**

기본 12-1 포인터를 사용하여 정수 합계를 구하는 예 — 12-1.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int aa[3];          ---- 정수형 배열을 선언한다.
06    int  *p;            ---- 정수형 포인터 변수를 선언한다.
07    int i, hap=0;
08
09    for(i=0; i < 3; i++)    ---- 배열에 숫자 3개를 입력한다.
10    {
11       printf(" %d 번째 숫자 : ", i+1);
12       scanf("%d", &aa[i]);
13    }
14
15    p = aa;             ---- 포인터 변수에 배열 aa의 주소를 대입한다.
16
17    for(i=0; i < 3; i++)    ---- 합계를 누적한다. aa[0]~aa[2]의 합계를 구한다.
18       hap = hap + *(p+i);
19
20    printf("입력 숫자의 합=> %d\n", hap);
21 }
```

1

2

실행 결과

```
1 번째 숫자 : 10
2 번째 숫자 : 20
3 번째 숫자 : 30
입력 숫자의 합=> 60
```

메모리 할당 함수
- 포인터 리뷰 (3/3)

- **예제 12-1 (2/2)**
  - ✓ 5행에서 정수형 배열 aa[3]을 선언하면 1031~1042번지에 4바이트×3 = 12바이트의 메모리 할당
  - ✓ 배열 aa는 1031번지 그 자체를 의미하는 포인터 상수

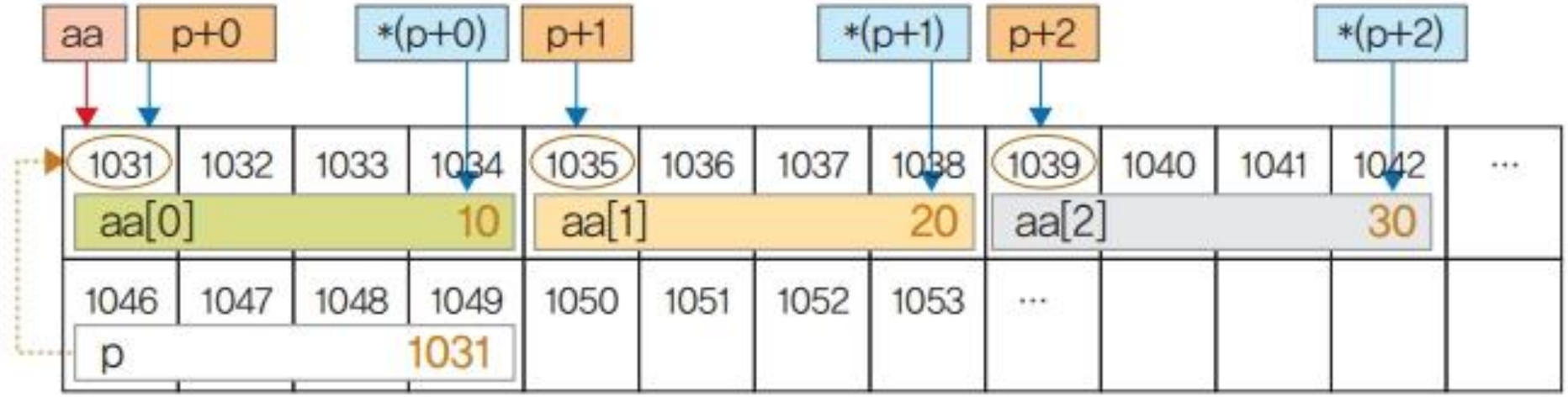

그림 12-1 배열과 포인터의 관계

메모리 할당 함수
– 동적 메모리 할당 필요성 (1/2)

- **동적 메모리 필요성**
  - ✓ 프로그램 실행 시 필요한 메모리 크기가 고정되는 경우 : 문제 없음
  - ✓ 프로그램 실행 시 필요한 메모리의 크기가 변하는 경우
    - ❶ 메모리 낭비 또는 메모리 부족
    - ❷ 사전에 정의한 메모리 크기로 프로그램의 처리 용량 제한
  - ✓ 해결 방법 : 필요 메모리 크기를 미리 정하지 않고, 필요할 때마다 확보
  - ➔ 동적 메모리 할당

메모리 할당 함수
– 동적 메모리 할당 필요성 (2/2)

- **동적 메모리 필요성 : 예제 12-2**

기본 12-2 고정된 크기의 배열로 인한 메모리 낭비의 예

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int aa[10000];          ---- 정수형 배열을 선언한다.
06    int  *p;                ---- 정수형 포인터 변수를 선언한다.
07    int i, hap=0;
08    int cnt;
09
10    printf(" 입력할 개수는 ? ");
11    scanf("%d", &cnt);      ---- 입력할 숫자의 개수를 입력한다.
12
13    for(i=0; i < cnt; i++)  ---- 입력한 개수만큼 배열에 숫자를 입력한다.
14    {
15       printf(" %d 번째 숫자 : ", i+1);
16       scanf("%d", &aa[i]);
17    }
18
19    p = aa;                 ---- 포인터 변수에 주소를 대입한다.
20
21    for(i=0; i < cnt; i++)  ---- 합계를 누적한다.
22       hap = hap + *(p+i);
23
24    printf("입력 숫자의 합 ==> %d\n", hap);  ---- 합계를 출력한다.
25 }
```

실행 결과

```
입력할 개수는 ? 3
1 번째 숫자 : 10
2 번째 숫자 : 20
3 번째 숫자 : 30
입력 숫자의 합 ==> 60
```

- **문제점 #1**

사용자 입력 숫자가 10000개 미만 시 메모리 공간 낭비 발생

- **문제점 #2**

10000개 이상 숫자 입력 시 메모리 부족 발생

- **문제점 #3**

프로그램의 최대 데이터 처리크기 (10000개) 한계가 있음

메모리 할당 함수
- malloc( ) (1/5)

- **예제 12-2 기준 메모리 낭비/부족을 막기 위해서는?**
  - ✓ 더하려는 숫자의 개수를 바탕으로 malloc( ) 함수를 사용하여 동적으로 메모리 확보
  - ✓ 만약 사용자가 숫자 3개를 입력하고자 한다면 메모리 세 칸을 확보하고 확보한 주소를 포인터 변수에 넣음

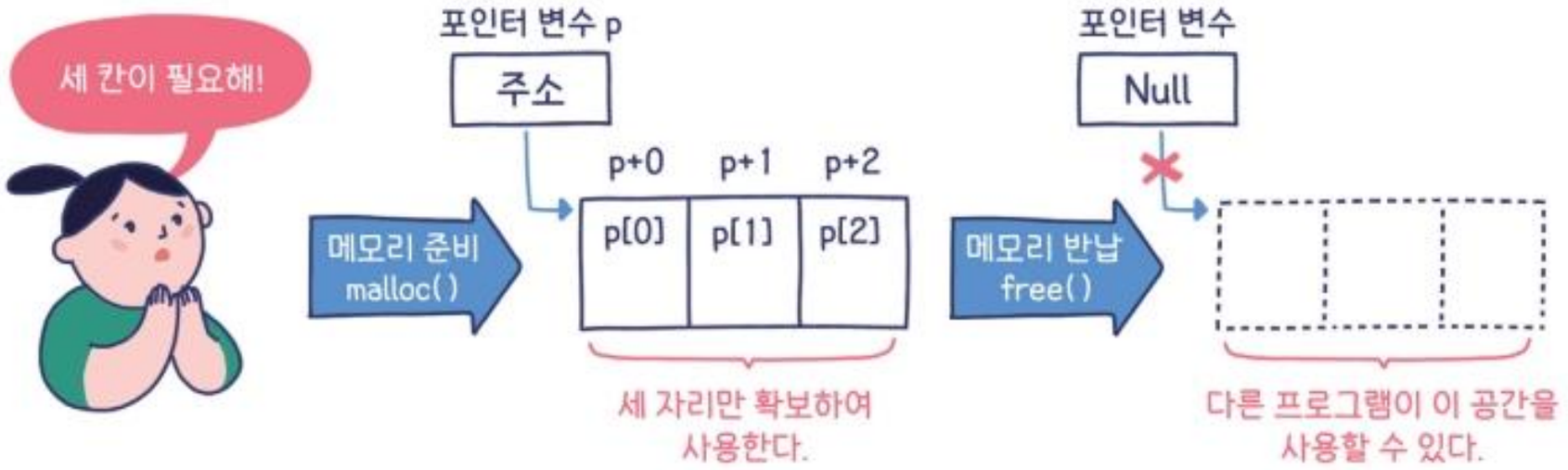

그림 12-3 동적 메모리 할당의 개념

메모리 할당 함수
– malloc( ) (2/5)

- **malloc( ) 함수의 사용 형식**

```
포인터 변수 = (포인터 변수의 데이터형*) malloc(포인터 변수의 데이터형 크기 × 필요한 크기)
```

  - ✓ 반환형은 void* 이며 원하는 데이터 타입으로 형변환해야 함
  - ✓ 매개변수는 할당하는 메모리 크기를 바이트 단위로 지정

- **malloc( ) 함수의 사용 예**
  - ✓ 포인터 변수를 int* p;로 선언한 경우

```
p = (int*) malloc(4 * 3);
```

  - ✓ int 형의 크기를 모를 경우, sizeof() 함수를 사용

```
p = (int*) malloc(sizeof(int) * 3);
```

메모리 할당 함수
– malloc( ) (3/5)

- **free() 함수 : 사용한 메모리를 반납**

  void free(void ptr);*

  ✓ 포인터 변수(ptr)에 널(null) 값을 넣어 준다는 의미

  - 포인터 변수는 아무것도 가리키지 않으므로, 이 공간을 운영체제에 반납

메모리 할당 함수
– malloc( ) (4/5)

- **예제 12-3 : 12-2를 malloc()를 이용하여 수정 (1/2)**

응용 12-3 malloc( ) 함수 사용 예 — 12-3.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <malloc.h>          ---- 메모리 관련 함수를 사용할 때 malloc.h를 추가해야 한다.
04 void main( )
05 {
06    int* p;                   ---- 정수형 포인터를 선언한다.
07    int i, hap=0;
08    int cnt;
09
10    printf(" 입력할 개수는 ? ");
11    scanf("%d", &cnt);        ---- 입력할 숫자의 개수를 입력한다.
12
13    p = (int*) __1__(sizeof(int) * cnt);   ---- 입력한 개수만큼 메모리를 확보한다.
14
15    for(i=0; i < cnt; i++)    ---- 입력한 개수(cnt)만큼 반복한다.
16    {
17       printf(" %d 번째 숫자 : ", i+1);
18       scanf("%d", __2__ );   ---- 공간이 확보된 포인터 변수 p에 입력받은 숫자를 입력한다. 배열처럼 &p[i]라고 입력해도 된다.
19    }
```

1

메모리 할당 함수
– malloc( ) (5/5)

- **예제 12-3 : 12-2를 malloc()를 이용하여 수정 (2/2)**

```
20
21    for(i=0; i < cnt; i++)
22       hap = hap + __3__          ----- 메모리의 실제 값을 합계에 누적한다. 배열처럼
23                                       p[i]라고 입력해도 된다.
24    printf("입력 숫자 합 ==> %d\n", hap);    ----- 합계를 출력한다.
25
26    free(p);                     ----- 메모리를 해제한다.
27 }
```

2

정답_ 1 malloc 2 p+i 3 *(p+i)

실행 결과

```
입력할 개수는 ? 3
1 번째 숫자 : 10
2 번째 숫자 : 30
3 번째 숫자 : 50
입력 숫자 합 ==> 90
```

메모리 할당 함수
- calloc( )

- **calloc() 함수 : 메모리를 할당하고 0으로 초기화**

  포인터 변수 = (포인터 변수의 데이터형*) calloc(필요한 크기, 포인터 변수의 데이터형 크기)

  ✓ malloc()과는 다르게 필요한 크기와 데이터형 크기를 각각 전달

메모리 할당 함수
- realloc( ) (1/5)

- **realloc() 함수 : 메모리 할당 크기를 실시간으로 변경**

```
포인터 변수 = (포인터 변수의 데이터형*) realloc(기본 포인터, 포인터 변수의 데이터형 크기 × 필요한 크기);
```

✓ 기본 포인터 : 크기를 변경하고자 하는 메모리를 가리키는 포인터

- **포인터 변수 p가 가리키는 메모리 크기를 10으로 변경하려면?**

```
p = (int*) realloc(p, sizeof(int) * 10);
```

✓ 반환값은 새로운 포인터 변수로 받아도 무방

메모리 할당 함수
– realloc( ) (2/5)

- **예제 : 사용자가 입력한 정수값들의 합계를 계산하여 출력**
  - ✓ 단, 사용자가 몇 개의 숫자를 입력할 지 사전에 정의하지 않으며 0을 입력하면 더 이상 입력을 받지 않음
- **malloc( )만을 사용할 경우**
  - ✓ 사용자에게 몇 개의 숫자를 입력할 지 먼저 입력을 받아야 함
  - ➔ 본 예제에는 적합하지 않음

메모리 할당 함수
– realloc( ) (3/5)

- **realloc( )을 사용할 경우**

  ❶ 최초 malloc( ) 함수로 메모리 한 칸을 확보, 사용자가 입력한 값을 넣음

  ❷ 다음 값을 입력받고 0이 아니면 realloc( ) 함수를 사용하여 크기를 늘려감

  ❸ 사용자가 0을 입력하면 필요한 작업을 한 후 free( ) 함수를 사용하여 메모리를 해제

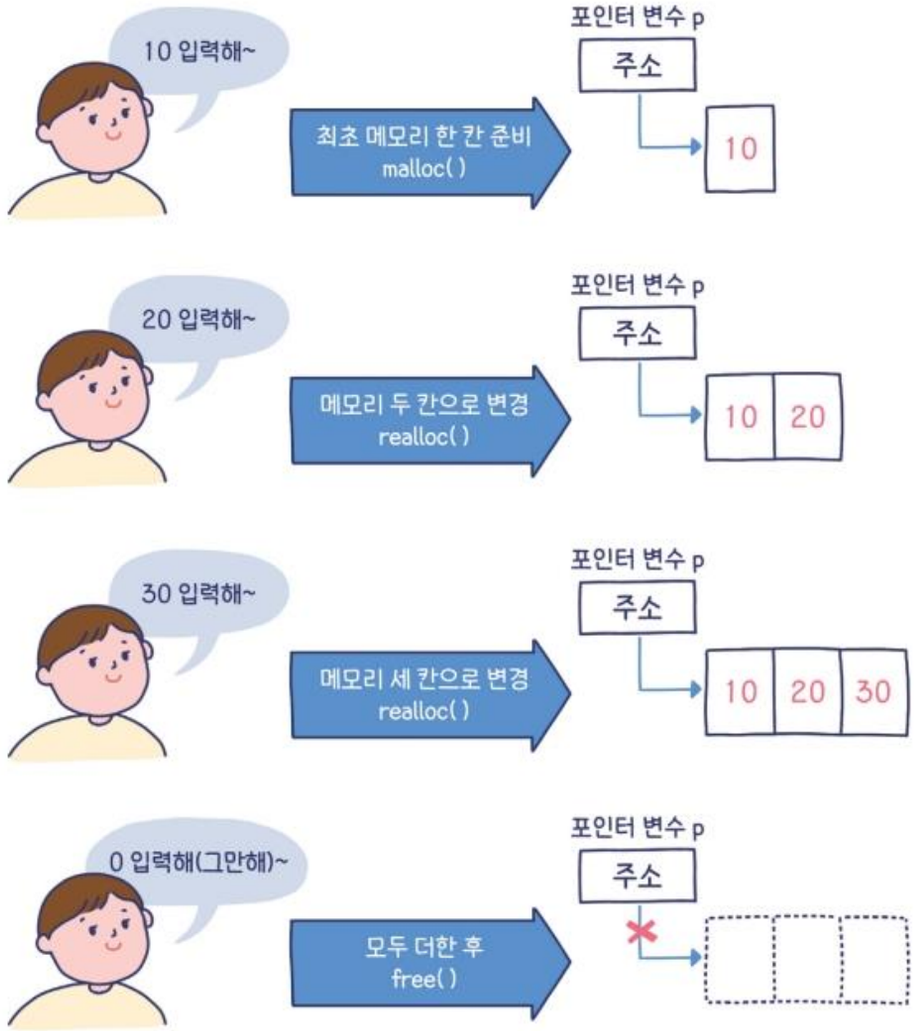

그림 12-4 realloc( ) 함수의 개념

메모리 할당 함수
– realloc( ) (4/5)

- **예제 12-5 (1/2) : 최초 입력 숫자는 0이 아니라고 가정할 때**

응용 12-5 realloc( ) 함수 사용 예 — 12-5.c

```
01  #define  _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03  #include <malloc.h>
04  void main( )
05  {
06     int* p;
07     int i, hap=0;
08     int cnt=0;
09     int data;
10
11     p = (int*) malloc(sizeof(int) * 1);
12     printf(" 1 번째 숫자 : ");
13     scanf("%d", &p[0]);
14     cnt++;
15
16     for(i=2;  ; i++)
17     {
18        printf(" %d 번째 숫자 : ", i);
19        scanf("%d", &data);
20
21        if(data != 0)
22           p = (int*) __1__(p, sizeof(int) * i);
23        else
24           __2__
25
26        p[i-1] = data;
27        cnt++;
28     }
29
30     for(i=0; i < cnt; i++)
31        hap = hap + p[i];
32
```

1

2

3

첫 번째 값을 입력받고 데이터 개수를 1 증가시킨다.

두 번째 값부터 계속 입력받는다. 조건이 없으므로 무한 루프가 된다.

바로 위에서 입력된 값을 임시 장소에 저장한다.

입력된 값이 0이 아니면 메모리를 한 칸 추가하고, 0이면 for문을 빠져나간다.

추가한 메모리 공간에 임시 장소의 값을 대입하고 입력값의 개수를 1 증가시킨다.

사용자가 입력한 개수(cnt)만큼 반복해서 합계를 구한다.

메모리 할당 함수
– realloc( ) (5/5)

- **예제 12-5 (2/2)**

```
33    printf("입력 숫자 합 ==> %d\n", hap);          ----- 합계를 출력한다.
34
35    free(p);                                      ----- 메모리를 해제한다.
36 }
```

정답_1 realloc 2 break;

실행 결과

```
1 번째 숫자 : 22
2 번째 숫자 : 45
3 번째 숫자 : 77
4 번째 숫자 : 0
입력 숫자 합 ==> 144
```

포인터 배열
– 다차원 배열 (1/3)

- **여러줄의 문자열을 저장할 경우**
  - ✓ 문자 하나만 저장 : char
  - ✓ 한 줄의 문자열 저장 : 배열 또는 포인터 변수 사용
  - ✓ 여러 줄의 문자열을 저장 : 다차원 배열

포인터 배열
– 다차원 배열 (2/3)

- **예제 12-6 : 문자열 반대로 출력하기 (1/2)**

기본 12-6 2차원 배열 사용 예 — 12-6.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char data[3][100];          ----- 3행 100열의 2차원 배열을 선언한다.
06    int i;
07
08    for(i=0; i < 3; i++)        ----- 세 번 반복한다.
09    {
10       printf(" %d 번째 문자열 : ", i+1);
11       gets(data[i]);           ------ 각 행에 최대 99자의 문자열을 입력한다.
12    }
13
14    printf("\n — 입력과 반대로 출력(이차원 배열) —\n");
15    for(i=2; i >= 0; i—)        ----- 2행, 1행, 0행의 순서로 마지막 행부터 출력한다.
16    {
17       printf(" %d :%s\n", i+1, data[i]);
18    }
19 }
```

실행 결과

```
1 번째 문자열 : Basic-C
2 번째 문자열 : Programming
3 번째 문자열 : Study

-- 입력과 반대로 출력(이차원 배열) --
3 :Study
2 :Programming
1 :Basic-C
```

3개의 문자열을 입력받고, 입력받은 반대 순서로 출력

포인터 배열
– 다차원 배열 (3/3)

- **예제 12-6 : 문자열 반대로 출력하기 (2/2)**
  - ✓ 사용자가 입력한 글자가 99자가 되지 않으면 낭비되는 공간이 너무 많음
  - ✓ 포인터 배열은 이런 공간 낭비의 단점을 극복하기 위한 것

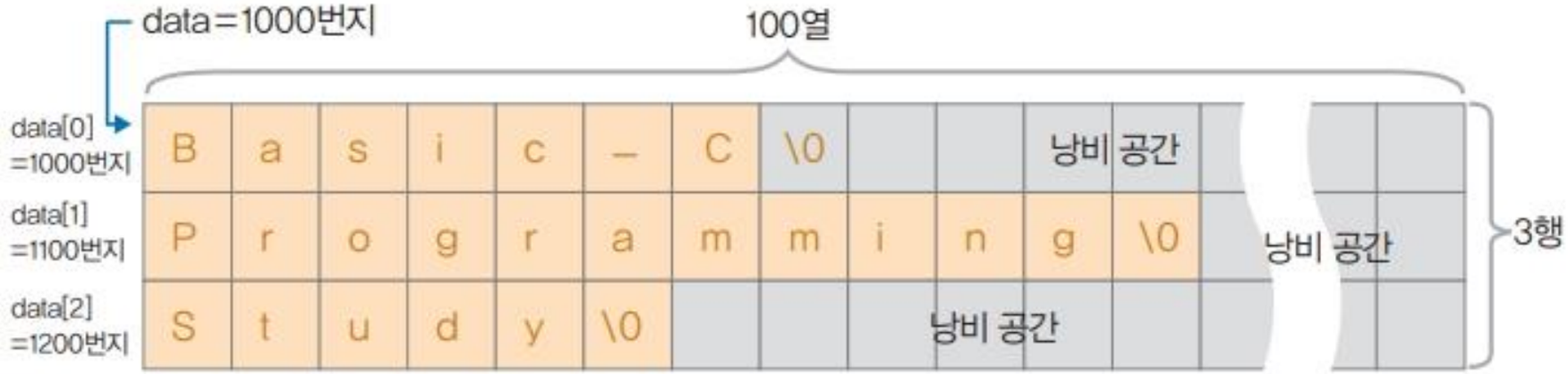

그림 12-5 2차원 배열의 메모리 낭비

포인터 배열
- 포인터 배열의 활용 (1/4)

- **포인터 배열 char* p[3] 선언**
  - ✓ 일반 배열처럼 p[0], p[1], p[2] 생성, 그 안에 주소 저장

그림 12-6 포인터 배열의 사용

- **일반 배열과 포인터 배열의 차이**
  - ✓ 일반 배열 : 정수 또는 문자가 들어감
  - ✓ 포인터 배열 : 주솟값이 들어감
    - 주솟값(예로 1000번지) 자체가 의미 있는 것이 아니라 그 주솟값이 가리키는 곳의 값이 중요함

포인터 배열
- 포인터 배열의 활용 (2/4)

- **일반 배열과 포인터 배열의 차이 (계속)**

그림 12-7 일반 배열과 포인터 배열의 비교

포인터 배열
- 포인터 배열의 활용 (3/4)

- **동적 메모리 할당과 포인터 배열을 이용한 예제 12-6의 개선 (1/2)**

응용 12-7 포인터 배열 사용 예 — 12-7.c

```
01 #include <stdio.h>
1  #include <malloc.h>
   #include <string.h>
04
05 void main( )
06 {
2     char* p[3];
      char  imsi[100];
09    int i, size;
10
11    for(i=0; i < 3; i++)
12    {
```

1 ---- 메모리 관련 함수와 문자열 관련 함수를 사용하기 위해 필요하다.

2 ---- 세 칸의 포인터 배열을 선언한다.

---- 입력값을 저장할 임시 공간 배열이다.

포인터 배열
- 포인터 배열의 활용 (4/4)

- **동적 메모리 할당과 포인터 배열을 이용한 예제 12-6의 개선 (2/2)**

```
13      printf(" %d 번째 문자열 : ", i+1);
14      gets(imsi);                          ----- 임시 공간에 문자열을 입력한다.
15
16      size = strlen(imsi);                 ----- 입력한 문자열의 길이를 계산한다.
17      p[i]  = (char*) malloc((sizeof(char) * size) + 1);   -----'입력한 길이+1' 크기의
                                                              메모리를 확보한다.
18
19      strcpy(__1__, imsi);                 ----- 입력한 문자열(imsi)의 내용을 메모리를 확보한
                                                   공간에 복사한다.
20    }
21
22    printf("\n -- 입력과 반대로 출력(포인터) --\n");
23    for(i=2; i >= 0; i--)
24    {
25      printf(" %d :%s\n", i+1, p[i]);      ----- 포인터 배열에 저장된 문자열을 출력한다.
26    }
27
28    for(i=0; i < 3; i++)                   ----- 할당했던 메모리 3개를 운영체제에 반납한다.
29      free(p[i]);
30  }
```

3 4 5

실행 결과

```
1 번째 문자열 : Basic-C
2 번째 문자열 : Programming
3 번째 문자열 : Study

-- 입력과 반대로 출력(포인터) --
3 :Study
2 :Programming
1 :Basic-C
```

실습

[실습 1] 사용자 입력 숫자 중 짝수만 더하기

예제 설명 사용자가 입력한 여러 숫자 중에서 짝수의 합계를 출력하는 프로그램이다([응용 12-3] 활용).

실행 결과

```
입력할 개수는 ? 4
1 번째 숫자 : 2
2 번째 숫자 : 40
3 번째 숫자 : 7
4 번째 숫자 : 11
입력한 짝수합 ==> 42
```

1. malloc() 함수를 사용
2. 예제 12-3의 코드를 활용

[실습 2] 사용자 입력 숫자 중 짝수만 더하기

1. 실습 1에서 사용자가 입력 개수를 입력하지 않는 경우에 대해 프로그램을 작성
2. 사용자는 처음부터 0을 입력할 수 있으며, 0을 입력 시 더 이상의 입력을 받지 않음
3. malloc()/realloc() 함수를 사용하라
4. 예제 12-5의 코드를 활용하라

[실습 3] 입력 문자열을 반대로 출력하기 (입력순서/글자순서)

예제 설명 입력한 순서의 반대로 그리고 각 행의 문자도 반대 순서로 출력하는 프로그램이다([응용 12-7] 활용).

실행 결과

```
1 번째 문자열 : IT CookBook
2 번째 문자열 : Basic C
3 번째 문자열 : Programming

-- 입력과 반대로 출력(포인터) : 글자 순서도 거꾸로 --
3 :gnimmargorP
2 :C cisaB
1 :kooBkooC TI
```

1. malloc() 함수를 사용하라
2. 포인터 배열을 사용하라
3. 예제 12-7의 코드를 활용하라

[실습 4] 사용자 문자열을 합하여 출력하기

1. 사용자로부터 2개의 문자열을 입력받음
2. 입력받은 2개의 문자열을 각각 포인터 배열에 저장
3. 2개의 문자열을 합치고(strcat()) 포인터 배열에 저장
4. 포인터 배열에 저장된 3개의 문자열을 출력
5. 본 프로그램에서는 동적 메모리 할당을 위해 malloc( )를 사용
6. 실습 3의 코드를 확장해서 구현

[실습 5] 블랙잭 프로그램 (1/3)

문제 설명

1. 블랙잭은 카드를 한장씩 받아 21에 가까운 수를 만드는 사람이 이기며 21을 초과하면 지는 게임
2. A는 1점, J/Q/K는 10점, 2~10은 숫자 그대로 점수를 부여함
3. 딜러는 합계 16점 이하에서는 반드시 1장의 카드를 더 받아야 하며, 17점 이상에서는 카드를 받지 않음
4. 유저는 최초 2장을 받고 카드를 더 받을 지 선택할 수 있음
5. 카드의 문양은 고려하지 않으며, 같은 숫자(& 알파벳)은 최대 4개까지 할당
6. 카드는 총 20장을 만들며 차례대로 유저와 딜러에게 나누어 줌

실행 결과

실행 결과 1.

```
Game Start? ABC
Error : Wrong Input, Try Again.
Game Start ? Start
Dealer : A, J
User : 3, K
More Card? Hit
Dealer : A, J
User : 3, K, Q
Busted! Over 21! Dealer Wins.
```

실행 결과 2.

```
Game Start ? Start
Dealer : A, 5
User : 5, J
More Card? Stay
Dealer : A, 5, 8
User : 5, J
Dealer : A, 5, 8, J
User : 5, J
User Wins! Dealer Busted!
```

실행 결과 3.

```
Game Start? Start
Dealer : A, 5
User : 2, K
More Card? Hit
Dealer : A, 5
User : 2, K, 8
More Card ? Stay
Dealer : A, 5, 4
User : 2, K, 8
Dealer : A, 5, 4, K
User : 2, K, 8
User Wins! Congratulation!
Game Start? End
```

실행 결과 4.

```
Game Start? Start
Dealer : J, 8
User : J, 9
More Card? Hit
Dealer : J, 8
User : J, 9, 2
BlackJack! Winner Winner Chicken
Dinner! User Win!
Game Start? End
```

실행 결과 5.

```
Game Start? Start
Dealer : J, 8
User : J, 9
More Card? Hit
Dealer : J, 8
User : J, 5, 3
More Card? Stay
Draw!
Game Start? End
```

[실습 5] 블랙잭 프로그램 (2/3)

코드 조건

1. 유저에게 Start, Quit 입력을 받아 게임 시작 및 종료 여부 받을 것.
2. 시작 카드는 유저, 딜러순으로 분배할 것.(시작카드도 유저, 딜러, 유저, 딜러 순서로)
3. 유저의 딜러의 카드는 카드 분배 후 언제나 공개할 것.
4. 카드를 받을 때마다 malloc을 통해서 추가로 할당하여 카드 받을 것.
5. 추가로 카드를 받을 시 Hit, 안받을 시 Stay를 입력받아 처리할 것.(유저가 원할 때 까지 카드를 받을 수 있음)
6. Stay를 할 경우, 딜러는 룰에 따라서 16점 이하일 경우 계속 카드를 받고 17점 이상일 시 더 이상 카드를 받지 않음.
7. 최종 합산이 21이 초과하지 않는 선에서 21에 가까운 사람이 승자.
8. 게임 종료 시, 다시 1번부터 시작

함수

1. main : 코드 내 조건문 처리(게임 시작)
2. Shuffle : 카드 20장 만드는 함수(malloc, random 사용)
3. Initialization : 카드를 2개 씩 사용자 딜러 순으로 분배하는 함수(malloc 사용)
4. Status : 현재 카드 상황을 출력하는 함수
5. Hit : 카드를 1장씩 유저에게 주는 함수(realloc 사용)
6. End : 딜러가 규칙에 따라 16점 이하일 경우 계속 카드를 받고, 17점 이상일 시 더이상 카드를 받지 않게 만드는 함수.
7. Result : 유저의 승리 및 패배를 검토하는 함수(free를 통한 메모리 해제)

[실습 5] 블랙잭 프로그램 (3/3)

- **1~100 난수(random) 발생 예제 : 다음 코드를 응용하여 구현**

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
        int i;
        srand(time(NULL));

        for (i = 1; i <= 10; i++)
                printf("%d ", (rand() % 100) + 1);
        printf("\n");
}
```

```
79 61 20 69 3 67 82 24 63 35
44 53 56 15 86 98 95 14 15 46
```

Q & A