사용자 정의 자료형

- typedef 선언, 중접 구조체, 열거형 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

구조체의 정의와 typedef 선언

◎ typedef 선언

- typedef 선언은 기존에 존재하는 자료형에 새로운 이름을 부여하는 선언

```
typedef   int   INT;
```

```
INT num;   // int num; 과 동일한 선언
INT * ptr;   // int * ptr; 과 동일한 선언
```

구조체의 정의와 typedef 선언

◎ TypeNameTypedef.c

```
typedef int INT;
typedef int * PTR_INT;

typedef unsigned int UINT;
typedef unsigned int * PTR_UINT;

typedef unsigned char UCHAR;
typedef unsigned char * PTR_UCHAR;

int main(void)
{
    INT num1 = 120;             // int num1 = 120;
    PTR_INT pnum1 = &num1;      // int * pnum1 = &num1;

    UINT num2 = 190;            // unsigned int num2 = 190;
    PTR_UINT pnum2 = &num2;     // unsigned int * pnum2 = &num2;

    UCHAR ch = 'Z';             // unsigned char ch = 'Z';
    PTR_UCHAR pch = &ch;        // unsigned char * pch = &ch;

    printf("%d, %u, %c \n", *pnum1, *pnum2, *pch);
    return 0;
}
```

```
120, 190, Z
```

새로 부여된 이름	대상 자료형
INT	int
PTR_INT	int *
UINT	unsigned int
PTR_UINT	unsigned int *
UCHAR	unsigned char
PTR_UCHAR	unsigned char *

구조체의 정의와 typedef 선언

◎ 구조체 정의와 typedef 선언

- 구조체 변수를 선언할 땐 반드시 struct 키워드를 사용해야함
- typedef 선언을 하면 struct 키워드를 사용하지 않는 형태로 구조체 변수를 선언할 수 있음

```
struct point
{
    int xpos;
    int ypos;
};

typedef struct point Point;
```

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;
```

- typedef 선언을 하면 구조체 이름은 사실 상 의미가 없기 때문에, typedef 선언 시 구조체 이름은 생략이 가능

```
typedef struct
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;
```

구조체의 정의와 typedef 선언

◎ StructTypedef.c

```
struct point
{
    int xpos;
    int ypos;
};

typedef struct point Point;

typedef struct person
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;

int main(void)
{
    Point pos={10, 20};
    Person man={"이승기", "010-1212-0001", 21};
    printf("%d %d \n", pos.xpos, pos.ypos);
    printf("%s %s %d \n", man.name, man.phoneNum, man.age);
    return 0;
}
```

```
10 20
이승기 010-1212-0001 21
```

함수로의 구조체 변수 전달과 반환

◎ 함수에서의 구조체 변수 사용

- 구조체 변수를 매개변수로 선언하여 함수 호출 시 인자로 전달할 수 있으며, 구조체 자료형을 반환값으로 사용할 수 있음

◎ StructValAndFunction.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

void ShowPosition(Point pos)
{
    printf("[%d, %d] \n", pos.xpos, pos.ypos);
}

Point GetCurrentPosition(void)
{
    Point cen;
    printf("Input current pos: ");
    scanf("%d %d", &cen.xpos, &cen.ypos);
    return cen;
}

int main(void)
{
    Point curPos=GetCurrentPosition();
    ShowPosition(curPos);
    return 0;
}
```

```
Input current pos: 2 4
[2, 4]
```

함수로의 구조체 변수 전달과 반환

◎ StructMemArrCopy.c

- 구조체 멤버에 배열이 선언되어도 인자로 전달하거나 반환할 수 있음

```
typedef struct person
{
    char name[20];
    char phoneNum[20];
    int age;
} Person;
void ShowPersonInfo(Person man)
{
    printf("name: %s \n", man.name);
    printf("phone: %s \n", man.phoneNum);
    printf("age: %d \n", man.age);
}

Person ReadPersonInfo(void)
{
    Person man;
    printf("name? "); scanf("%s", man.name);
    printf("phone? "); scanf("%s", man.phoneNum);
    printf("age? "); scanf("%d", &man.age);
    return man;
}
```

```
int main(void)
{
    Person man=ReadPersonInfo();
    ShowPersonInfo(man);
    return 0;
}
```

```
name? Jung
phone? 010-12XX-34XX
age? 22
name: Jung
phone: 010-12XX-34XX
age: 22
```

함수로의 구조체 변수 전달과 반환

◎ 구조체 변수를 대상으로 하는 call-by-reference

- StructFunctionCallByRef.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

void OrgSymTrans(Point * ptr)    // 원점대칭
{
    ptr->xpos = (ptr->xpos) * -1;
    ptr->ypos = (ptr->ypos) * -1;
}

void ShowPosition(Point pos)
{
    printf("[%d, %d] \n", pos.xpos, pos.ypos);
}

int main(void)
{
    Point pos={7, -5};
    OrgSymTrans(&pos);    // pos의 값을 원점 대칭이동시킨다.
    ShowPosition(pos);
    OrgSymTrans(&pos);    // pos의 값을 원점 대칭이동시킨다.
    ShowPosition(pos);
    return 0;
}
```

```
[-7, 5]
[7, -5]
```

함수로의 구조체 변수 전달과 반환

◎ 구조체 변수를 대상으로 한 연산

- StructOperation.c
  - 대입 연산, & 연산, sizeof 연산 정도만 제한적으로 허용

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

int main(void)
{
    Point pos1={1, 2};
    Point pos2;
    pos2=pos1;  // pos1의 멤버 대 pos2의 멤버간 복사가 진행됨

    printf("크기: %d \n", sizeof(pos1));   // pos1의 전체 크기 반환
    printf("[%d, %d] \n", pos1.xpos, pos1.ypos);
    printf("크기: %d \n", sizeof(pos2));   // pos2의 전체 크기 반환
    printf("[%d, %d] \n", pos2.xpos, pos2.ypos);
    return 0;
}
```

```
크기: 8
[1, 2]
크기: 8
[1, 2]
```

함수로의 구조체 변수 전달과 반환

◎ 구조체 변수를 대상으로 한 연산

- 산술 연산은 정의되어 있지 않기 때문에 개발자가 별도의 함수로 구현해야 함
- StructAddMin.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

Point AddPoint(Point pos1, Point pos2)
{
    Point pos={pos1.xpos+pos2.xpos, pos1.ypos+pos2.ypos};
    return pos;
}

Point MinPoint(Point pos1, Point pos2)
{
    Point pos={pos1.xpos-pos2.xpos, pos1.ypos-pos2.ypos};
    return pos;
}
```

```
int main(void)
{
    Point pos1={5, 6};
    Point pos2={2, 9};
    Point result;

    result=AddPoint(pos1, pos2);
    printf("[%d, %d] \n", result.xpos, result.ypos);
    result=MinPoint(pos1, pos2);
    printf("[%d, %d] \n", result.xpos, result.ypos);
    return 0;
}
```

```
[7, 15]
[3, -3]
```

구조체의 유용함에 대한 논의와 중첩 구조체

◎ 구조체를 정의하는 이유

- 구조체를 통해 연관 있는 데이터를 하나로 묶을 수 있는 자료형을 정의하면, 데이터의 표현 및 관리가 용이해지고, 합리적인 코드를 작성할 수 있게 함
  - StructImportant.c

```
typedef struct student
{
    char name[20];      // 학생 이름
    char stdnum[20];    // 학생 고유번호
    char school[20];    // 학교 이름
    char major[20];     // 선택 전공
    int year;           // 학년
} Student;

void ShowStudentInfo(Student * sptr)
{
    printf("학생 이름: %s \n", sptr->name);
    printf("학생 고유번호: %s \n", sptr->stdnum);
    printf("학교 이름: %s \n", sptr->school);
    printf("선택 전공: %s \n", sptr->major);
    printf("학년: %d \n", sptr->year);
}
```

```
int main(void)
{
    Student arr[7];
    int i;

    for(i=0; i<7; i++)
    {
        printf("이름: "); scanf("%s", arr[i].name);
        printf("번호: "); scanf("%s", arr[i].stdnum);
        printf("학교: "); scanf("%s", arr[i].school);
        printf("전공: "); scanf("%s", arr[i].major);
        printf("학년: "); scanf("%d", &arr[i].year);
    }

    for(i=0; i<7; i++)
        ShowStudentInfo(&arr[i]);
    return 0;
}
```

  - 구조체를 선언함으로써 함수 호출 시 데이터를 간결하게 전달하고, 배열을 사용하여 비슷한 데이터 묶음을 효율적으로 처리

구조체의 유용함에 대한 논의와 중첩 구조체

◎ 중첩 구조체

- 먼저 정의한 구조체를 다른 구조체의 멤버로 선언하여 구현
  - CircleIncludePoint.c

```
typedef struct point
{
    int xpos;
    int ypos;
} Point;

typedef struct circle
{
    Point cen;
    double rad;
} Circle;
```

```
void ShowCircleInfo(Circle * cptr)
{
    printf("[%d, %d] \n", (cptr->cen).xpos, (cptr->cen).ypos);
    printf("radius: %g \n\n", cptr->rad);
}

int main(void)
{
    Circle c1={{1, 2}, 3.5};
    Circle c2={2, 4, 3.9};
    ShowCircleInfo(&c1);
    ShowCircleInfo(&c2);
    return 0;
}
```

```
[1, 2]
radius: 3.5

[2, 4]
radius: 3.9
```

  - 중괄호를 이용해 구조체 변수를 구분하여 초기화 하지 않으면, 선언된 순서에 따라 변수가 초기화 됨

열거형의 정의와 의미

◎ 열거형(Enumerated Type)의 정의와 열거형 변수의 선언

- 의미 있는 상수의 정의를 통한 의미 부여 방법
- 열거형의 정의

```
enum syllable     // syllable이라는 이름의 열거형 정의
{
     Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7
};
```

- 열거형 변수의 선언

```
enum syllable tone;     // syllable형 변수 tone의 선언
```

열거형의 정의와 의미

◎ EnumTypeTone.c

```
typedef enum syllable
{
    Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7
} Syllable;
void Sound(Syllable sy)
{
    switch(sy)
    {
    case Do:
        puts("도는 하얀 도라지 ♪"); return;
    case Re:
        puts("레는 둥근 레코드 ♩"); return;
    case Mi:
        puts("미는 파란 미나리 ♩♪"); return;
    case Fa:
        puts("파는 예쁜 파랑새 ♪♭"); return;
    case So:
        puts("솔은 작은 솔방울 ♩♪♪"); return;
    case La:
        puts("라는 라디오고요~ ♪♩♭♩"); return;
    case Ti:
        puts("시는 졸졸 시냇물 ♩♭♩♪"); return;
    }
    puts("다 함께 부르세~ 도레미파 솔라시도 솔 도~ 짠~");
}
```

```
int main(void)
{
    Syllable tone;
    for(tone=Do; tone<=Ti; tone+=1)
        Sound(tone);
    return 0;
}
```

```
도는 하얀 도라지 ♪
레는 둥근 레코드 ♩
미는 파란 미나리 ♩♪
파는 예쁜 파랑새 ♪♭
솔은 작은 솔방울 ♩♪♪
라는 라디오고요~ ♪♩♭♩
시는 졸졸 시냇물 ♩♭♩♪
```

열거형의 정의와 의미

◎ 열거형 상수의 값 배정

- 열거형 상수의 값이 명시되지 않은 경우는 0부터 시작하여 1씩 증가한 정수를 배정

```
enum color {RED, BLUE, WHITE, BLACK};
```

동일한 선언

```
enum color {RED=0, BLUE=1, WHITE=2, BLACK=3};
```

- 중간에 값을 생략하면 앞서 정의한 상수에 1을 증가시켜 배정

```
enum color {RED=3, BLUE, WHITE=6, BLACK};
```

동일한 선언

```
enum color {RED=3, BLUE=4, WHITE=6, BLACK=7};
```

열거형의 정의와 의미

◎ 열거형의 활용

- 둘 이상의 연관 있는 이름을 상수로 선언하여 프로그램의 가독성을 높임
- 새로운 상수 자료형 정의

```
typedef enum syllable
{
    Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7
} Syllable;
```

- 열거형 이름을 생략하여 열거형을 정의하기도 함 (상수 선언과 동일)

```
enum {
   Do=1, Re=2, Mi=3, Fa=4, So=5, La=6, Ti=7 };
```