# 14 Structures and Other Data Forms

### Keywords: struct , union , typedef

### Operators: . ->

### C structures는 무엇이고 structure templates과 variables을 만드는 방법

### structure의 member를 접근하는 방법, structures를 다루는 function을 작성하는 방법

### C의 typedef 기능

### Unions과 pointers to functions

#### program을 설계하는 데 있어서 가장 중요한 steps 중의 하나: data를 표현하는 좋은 방법을 선택하는 것

##### C structure variables: data을 더 좋은 방법으로 표현하는 방법을 제공

## 14.1 Sample Problem: Creating an Inventory of Books

### struct위한 format 또는 layout를 설정

### structure layout을 위한 변수를 선언

### structure variable의 개별 component를 접근

// Listing 14.1 The book.c Program

//\* book.c -- one-book inventory \*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

char\* s\_gets(char\* st, int n);

#define MAXTITL 41 /\* maximum length of title + 1 \*/

#define MAXAUTL 31 /\* maximum length of author's name + 1 \*/

struct book { /\* structure template: tag is book \*/

char title[MAXTITL];

char author[MAXAUTL];

float value;

}; /\* end of structure template \*/

int main(void)

{

struct book library; /\* declare library as a book variable \*/

printf("Please enter the book title.\n");

s\_gets(library.title, MAXTITL); /\* access to the title portion \*/

printf("Now enter the author.\n");

s\_gets(library.author, MAXAUTL);

printf("Now enter the value.\n");

scanf("%f", &library.value);

printf("%s by %s: $%.2f\n", library.title,

library.author, library.value);

printf("%s: \"%s\" ($%.2f)\n", library.author,

library.title, library.value);

printf("Done.\n");

system("pause");

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

char\* find;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

find = strchr(st, '\n'); // look for newline

if (find) // if the address is not NULL,

\*find = '\0'; // place a null character there

else

while (getchar() != '\n')

continue; // dispose of rest of line

}

return ret\_val;

}

## 14.2 Structure 선언

### A structure declaration은 structure에 같이 표현할 변수들을 기술하는 것

**struct book {**

**char title[MAXTITL];**

**char author[MAXAUTL];**

**float value;**

**};**

### structure 선언을 template이라 한다. 그 이유는 data를 어떻게 저장할 것인지에 대한 outline을 기술하기 때문이다

### struct book library;

#### keyword struct: 다음에 오는 것은 structure을 기술한다는 것을 의미한다

#### tag: shorthand label *book*

##### *library*는 structure variable

## 14.3 Defining a Structure Variable

### “structure plan”

#### compiler에게 data 를 어떻게 표현하는 지를 알려준다

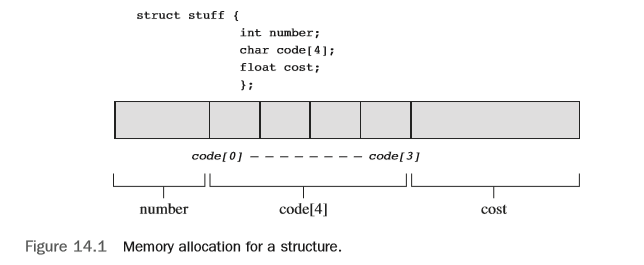
#### data를 위한 메모리 공간을 할당하는 것은 아니다

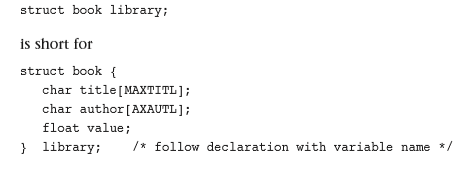
### structure variable의 생성

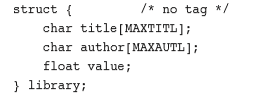
#### book template를 사용하여 메모리 공간을 할당

struct book doyle, panshin, \* ptbook;

###### panshin = doyle;







## 14.3.1 Structure 초기화

### variable과 array의 초기화

int count = 0;

int fibo[7] = {0,1,1,2,3,5,8};

### structure의 초기화

#### comma를 사용- braces 안에 초기화 값의 list.

**struct book library = {**

**"The Pious Pirate and the Devious Damsel",**

**"Renee Vivotte",**

**1.95**

**};**

## 14.3.2 Structure Members의 접근

### structure의 개별 member를 접근하는 방법

#### dot(.)를 사용, structure member operator라고 부름

**s\_gets(library.title, MAXTITL);**

**scanf("%f", &library.value);**

## 14.4 Arrays of Structures

### 두 권의 books을 기술하기 위해, 두개의 변수를 사용하는 방법

### n개의 books을 기술하기 위해, array of structures를 사용

// Listing 14.2 The manybook.c Program

/\* manybook.c -- multiple book inventory \*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

char\* s\_gets(char\* st, int n);

#define MAXTITL 40

#define MAXAUTL 40

#define MAXBKS 100 /\* maximum number of books \*/

struct book { /\* set up book template \*/

char title[MAXTITL];

char author[MAXAUTL];

float value;

};

int main(void)

{

struct book library[MAXBKS]; /\* array of book structures \*/

int count = 0;

int index;

printf("Please enter the book title.\n");

printf("Press [enter] at the start of a line to stop.\n");

while (count < MAXBKS && s\_gets(library[count].title, MAXTITL) != NULL

&& library[count].title[0] != '\0')

{

printf("Now enter the author.\n");

s\_gets(library[count].author, MAXAUTL);

printf("Now enter the value.\n");

scanf("%f", &library[count++].value);

while (getchar() != '\n')

continue; /\* clear input line \*/

if (count < MAXBKS)

printf("Enter the next title.\n");

}

if (count > 0)

{

printf("Here is the list of your books:\n");

for (index = 0; index < count; index++)

printf("%s by %s: $%.2f\n", library[index].title,

library[index].author, library[index].value);//struct 배열 사용 예제

}

else

printf("No books? Too bad.\n");

system("pause");

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

char\* find;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

find = strchr(st, '\n'); // look for newline

if (find) // if the address is not NULL,

\*find = '\0'; // place a null character there

else

while (getchar() != '\n')

continue; // dispose of rest of line

}

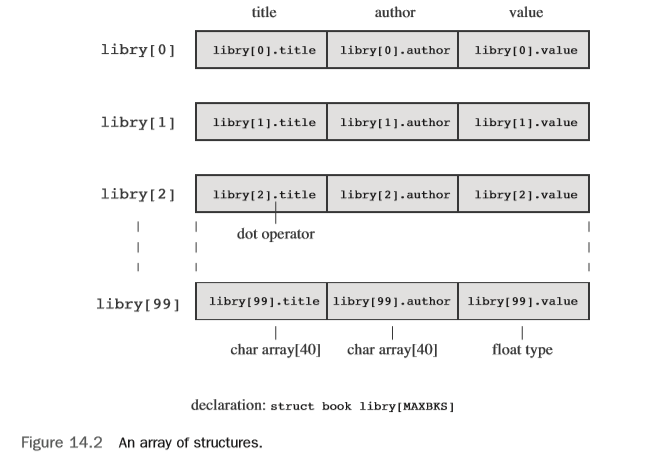
return ret\_val;

}

* arrays of structures를 선언하는 방법과 개별 members를 접근하는 방법.

## 14.4.1 Array of Structures의 선언

struct book library[MAXBKS];



## 14.4.2 Array of Structures의 member를 식별하는 방법

### array of structures의 member를 식별하는 방법

#### structure name + dot operator + member name

library[0].value /\* the value associated with the first array element \*/

library[4].title /\* the title associated with the fifth array element \*/

library.value[2] // WRONG

library[2].value // RIGHT

library // an array of book structures

library[2] // an array element, hence a book structure

library[2].title // a char array (the title member of library[2])

library[2].title[4] // a char in the title member array

## 14.5 Nested Structures

// Listing 14.3 The friend.c Program

// friend.c -- example of a nested structure

#include <stdio.h>

#include <stdlib.h>

#define LEN 20

const char\* msgs[5] =

{

" Thank you for the wonderful evening, ",

"You certainly prove that a ",

"is a special kind of guy. We must get together",

"over a delicious ",

" and have a few laughs"

};

struct names { // first structure

char first[LEN];

char last[LEN];

};

struct guy { // second structure

struct names handle; // nested structure

char favfood[LEN];

char job[LEN];

float income;

};

int main(void)

{

struct guy fellow = { // initialize a variable

{ "Ewen", "Villard" },

"grilled salmon",

"personality coach",

68112.00

};

printf("Dear %s, \n\n", fellow.handle.first);

printf("%s%s.\n", msgs[0], fellow.handle.first);

printf("%s%s\n", msgs[1], fellow.job);

printf("%s\n", msgs[2]);

printf("%s%s%s", msgs[3], fellow.favfood, msgs[4]);

if (fellow.income > 150000.0)

puts("!!");

else if (fellow.income > 75000.0)

puts("!");

else

puts(".");

printf("\n%40s%s\n", " ", "See you soon,");

printf("%40s%s\n", " ", "Shalala");

system("pause");

return 0;

}

struct names { // first structure

char first[LEN];

char last[LEN];

};

struct guy { // second structure

struct names handle; // nested structure

char favfood[LEN];

char job[LEN];

float income;

};

int main(void)

{

struct guy fellow = { // initialize a variable

{ "Ewen", "Villard" },

"grilled salmon",

"personality coach",

68112.00

};

printf("Hello, %s!\n", fellow.handle.first);

* (fellow.handle).first

## 14.6 Pointers to Structures

### pointers to structures를 사용하는 것이 good idea인 4가지 이유

#### pointers to structures는 structure 그 자체보다 조작이 더 쉽다

#### structure가 argument to a function로서 전달되는 것은 값 복사에 의한 방법으로 pointer to a structure으로 전달이 더 효율적이다

#### structure가 배열일때는 pointer를 전달하는 방법이 더 효율적이다

#### pointers to structures를 포함하는 structure를 사용할 수 있다.

### pointer to a structure를 정의하는 방법, structure member를 접근하기 위해 pointer를 사용하는 방법

// Listing 14.4 The friends.c Program

/\* friends.c -- uses pointer to a structure \*/

#include <stdio.h>

#include <stdlib.h>

#define LEN 20

struct names {

char first[LEN];

char last[LEN];

};

struct guy {

struct names handle;

char favfood[LEN];

char job[LEN];

float income;

};

int main(void)

{

struct guy fellow[2] = {

{ { "Ewen", "Villard" },

"grilled salmon",

"personality coach",

68112.00

},

{ { "Rodney", "Swillbelly" },

"tripe",

"tabloid editor",

232400.00

}

};

struct guy\* him; /\* here is a pointer to a structure \*/

printf("address #1: %p, %p, %p #2: %p\n", fellow, &fellow, &fellow[0], &fellow[1]);

him = &fellow[0]; /\* tell the pointer where to point \*/

printf("pointer #1: %p #2: %p\n", him, him + 1);//him+1의 주소가 어디?

printf("him->income is $%.2f: (\*him).income is $%.2f\n",

him->income, (\*him).income);

him++; /\* point to the next structure \*/

printf("him->favfood is %s: him->handle.last is %s\n",

him->favfood, him->handle.last);

system("pause");

return 0;

}

## 14.6.1 Structure Pointer를 선언하고 초기화하는 방법

struct guy \* him;

### 이러한 선언은 new structure를 만들지 않는다

#### pointer him 은 guy type의 어떤 structure도 point할 수 있다

##### barney가 guy type의 structure라면

him = &barney;

##### structure의 이름이 structure의 address가 아니다; & operator를 사용하는 것이 필요하다

##### fellow가 array of structures라면

him = &fellow[0];

###### fellow는 array of structures이므로, fellow[0]는 structure이다

#### him이 point to fellow[0]을 가리키도록 초기화한다

##### him points to fellow[0]

###### him은 배열에 대한 pointer이다

##### him + 1 points to fellow[1] .

## 14.6.2 Member Access by Pointer

### pointer him은 structure fellow[0]를 point한다.

### operator ->를 사용한다

### fellow[0]의 member 값을 가져오기 위해 pointer him를 사용하는 방법

#### him->income는 barney.income를 의미함 if him == &barney

#### him->income는 fellow[0].income를 의미함 if him == &fellow[0]

### him 은 pointer임, him->income 은 pointed-to structure의 member임

### if him == &fellow[0] , then \*him == fellow[0]

fellow[0].income == (\*him).income

### him이 barney라는 type guy structure에 대한 pointer이면 다음은 모두 같다

#### . operator는 \* 연산자보다 우선 순위가 높다

barney.income == (\*him).income == him->income // assuming him == &barney

## 14.7 Functions에 Structures 전달

### function arguments는 값을 function에 전달하는 것이 가장 기본적인 처리 방법

#### passing structures as arguments

#### passing pointers to structures as arguments

## 14.7.1 Passing Structure Members

// Listing 14.5 The funds1.c Program

/\* funds1.c -- passing structure members as arguments \*/

#include <stdio.h>

#include <stdlib.h>

#define FUNDLEN 50

struct funds {

char bank[FUNDLEN];

double bankfund;

char save[FUNDLEN];

double savefund;

};

double sum(double, double);

int main(void)

{

struct funds stan = {

"Garlic-Melon Bank",

4032.27,

"Lucky's Savings and Loan",

8543.94

};

printf("Stan has a total of $%.2f.\n",

sum(stan.bankfund, stan.savefund));

printf(" bankfund = %f\n", stan.bankfund);

system("pause");

return 0;

}

/\* adds two double numbers \*/

double sum(double x, double y)

{

return(x++ + y++);

}

* pass structure members as a function arguments

## 14.7.2 Structure Address를 argument로 전달하는 방법

### structure의 address를 argument로서 전달하는 방법

// Listing 14.6 The funds2.c Program

/\* funds2.c -- passing a pointer to a structure \*/

#include <stdio.h>

#include <stdlib.h>

#define FUNDLEN 50

struct funds {

char bank[FUNDLEN];

double bankfund;

char save[FUNDLEN];

double savefund;

};

//double sum(const struct funds); /\* argument is structure \*/

double sum(struct funds\* f); /\* argument is a pointer \*/

int main(void)

{

struct funds stan = {

"Garlic-Melon Bank",

4032.27,

"Lucky's Savings and Loan",

8543.94

};

//printf("Stan has a total of $%.2f.\n", sum(stan));

printf("Stan has a total of $%.2f.\n", sum(&stan));

system("pause");

return 0;

}

//double sum(const struct funds money)

double sum(struct funds\* money)

{

/\*

printf("sum::ban = %f\n", money.bankfund);

return(money.bankfund + money.savefund);

\*/

money->bankfund += 1000.00;

printf("sum::ban = %f\n", money->bankfund);

return(money->bankfund + money->savefund);

}

### structure’s address를 전달하기 위해 & operator를 사용해야 한다.

#### array name과 달리, structure name 그 자체는 address가 아니다

#### passing the address &stan to the function

##### pointer money to point to the structure stan .

## 14.7.3 Structure를 Argument로서 전달하는 방법

// Listing 14.7 The funds3.c Program

/\* funds3.c -- passing a structure \*/

#include <stdio.h>

#include <stdlib.h>

#define FUNDLEN 50

struct funds {

char bank[FUNDLEN];

double bankfund;

char save[FUNDLEN];

double savefund;

};

double sum(struct funds moolah); /\* argument is a structure \*/

int main(void)

{

struct funds stan = {

"Garlic-Melon Bank",

4032.27,

"Lucky's Savings and Loan",

8543.94

};

printf("Stan has a total of $%.2f.\n", sum(stan));

printf("main()::stan.bankfund = %f\n", stan.bankfund);

system("pause");

return 0;

}

double sum(struct funds moolah)

{

moolah.bankfund += 1000.00;

return(moolah.bankfund + moolah.savefund);

}

### sum()이 호출될 때, automatic variable called moolah이 생성된다.변수 구조는 funds template로 만들어진다

#### structure members는 값 복사로 초기화된다

## 14.7.4 Structure type의 호환성

### n\_data과 o\_data 이 same type의 structure이면 치환 가능

o\_data = n\_data; // assigning one structure to another

### 한 structure의 초기화는 same type의 다른 structure로 가능

struct names right\_field = {"Ruthie", "George"};

struct names captain = right\_field; // initialize a structure to another

### Using structures as function arguments

### using functions to return structures

#### Structure pointers는 two-way communication 가능

##### function arguments

##### return type of function //listing 14.9

// Listing 14.8 The names1.c Program

/\* names1.c -- uses pointers to a structure \*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define NLEN 30

struct namect {

char fname[NLEN];

char lname[NLEN];

int letters;

};

void getinfo(struct namect\*);

void makeinfo(struct namect\*);

void showinfo(const struct namect\*);

char\* s\_gets(char\* st, int n);

int main(void)

{

struct namect person;

getinfo(&person);

makeinfo(&person);

showinfo(&person);

system("pause");

return 0;

}

void getinfo(struct namect\* pst)

{

printf("Please enter your first name.\n");

s\_gets(pst->fname, NLEN);

printf("Please enter your last name.\n");

s\_gets(pst->lname, NLEN);

}

void makeinfo(struct namect\* pst)

{

pst->letters = strlen(pst->fname) +

strlen(pst->lname);

}

void showinfo(const struct namect\* pst)

{

printf("%s %s, your name contains %d letters.\n",

pst->fname, pst->lname, pst->letters);

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

char\* find;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

find = strchr(st, '\n'); // look for newline

if (find) // if the address is not NULL,

\*find = '\0'; // place a null character there

else

while (getchar() != '\n')

continue; // dispose of rest of line

}

return ret\_val;

}

* using structure arguments and return values.
* // Listing 14.9 The names2.c Program
* /\* names2.c -- passes and returns structures \*/
* // coding 능력이 있어야 함
* #include <stdio.h>
* #include <string.h>
* #include <stdlib.h>
* #define NLEN 30
* struct namect {
* char fname[NLEN];
* char lname[NLEN];
* int letters;
* };
* struct namect getinfo(void);
* struct namect makeinfo(struct namect);
* void showinfo(struct namect);
* char\* s\_gets(char\* st, int n);
* int main(void)
* {
* struct namect person;
* person = getinfo();
* person = makeinfo(person);
* showinfo(person);
* system("pause");
* return 0;
* }
* struct namect getinfo(void)
* {
* struct namect temp;
* printf("Please enter your first name.\n");
* s\_gets(temp.fname, NLEN);
* printf("Please enter your last name.\n");
* s\_gets(temp.lname, NLEN);
* return temp;
* }
* struct namect makeinfo(struct namect info)
* {
* info.letters = strlen(info.fname) + strlen(info.lname);
* return info;
* }
* void showinfo(struct namect info)
* {
* printf("%s %s, your name contains %d letters.\n",
* info.fname, info.lname, info.letters);
* }
* char\* s\_gets(char\* st, int n)
* {
* char\* ret\_val;
* char\* find;
* ret\_val = fgets(st, n, stdin);
* if (ret\_val)
* {
* find = strchr(st, '\n'); // look for newline
* if (find) // if the address is not NULL,
* \*find = '\0'; // place a null character there
* else
* while (getchar() != '\n')
* continue; // dispose of rest of line
* }
* return ret\_val;
* }

## 14.7.5 Structures를 사용할 것인가? 아니면 Pointer to Structures?

### structure pointers를 arguments로 사용할 것인가? 아니면 structure 그 자체를 arguments와 return values로 사용할 것인가?

### passing structures as arguments의 장점

#### 호출 function은 복사 값으로 처리

##### vector ans, vectors a and b를 structure 변수로 선언

struct vector {double x; double y;};

struct vector ans, a, b;

struct vector sum\_vect(struct vector, struct vector);

...

ans = sum\_vect(a,b);

### pointer 사용시

struct vector ans, a, b;

void sum\_vect(const struct vector \*, const struct vector \*, struct vector \*);

* ...

sum\_vect(&a, &b, &ans);

### function arguments로서 structure pointers를 사용

#### 사용 이유: 효율성

## 14.7.6 structure 내부에서 Character Arrays 또는 Character Pointers 사용

## 14.7.6.1 character string의 scanf() 사용

### structure 내부에 character arrays를 사용하는 방법

char name[40]; scanf("%s", name);

char \*name; scanf("%s", name); // error, 차이를 이해하는 것이 중요

#define LEN 20

struct names {

char first[LEN];

char last[LEN];

};

struct pnames {

char \* first;

char \* last;

};

### struct pnames은 strings를 저장할 메모리를 할당하지 않는다

#### struct pnames내부에 있는 first, last pointer는 string를 가리키는 pointer이다

##### string은 program의 데이터 영역의 읽기 전용 초기화 영역에 만들어짐

struct names veep = {"Talia", "Summers"};

struct pnames treas = {"Brad", "Fallingjaw"}; //가능하다

struct pnames t;

t.first = "hong";

printf("%s and %s\n", veep.first, treas.first);

#### struct names veep은 해당 strings이 structure 내부에 복사 저장된다

#### struct pnames treas:

##### strings은 string constants로서 데이터 영역의 읽기 전용 초기화 영역에 있다

##### struct pnames은 strings를 저장하는 공간을 갖고 있지 않고 pointer만 저장한다

###### pnames structure 내부의 pointer는 compiler가 생성한 string constant를 가리킨다

###### struct 변수 attorney에 대하여, scanf()는 attorney.last로 지정된 address에 string을 입력받아 저장하게 한다. attorney.last는 uninitialized variable이므로, scanf()가 그 address에 입력된 string을 복사할 때 오류가 발생한다

struct names accountant;

struct pnames attorney;

puts("Enter the last name of your accountant:");

scanf("%s", *accountant.last);*

puts("Enter the last name of your attorney:");

scanf("%s", attorney.last); /\* here lies the danger \*/

### pointers-to-char로 struct member를 저장하고자 하나, serious misuse가 될 잠재적 가능성이 매우 크다

#### structure에 strings을 member 값으로 저장하고자 할 때는 , character array members를 사용하는 것이 간단하고 안전하다

## 14.7.7 Structure, Pointers, and malloc()

### string을 다루기 위해 struct 내에 pointer를 사용

#### memory를 할당하기 위한 malloc()를 사용하고 그 address를 저장하기 위하여 pointer를 사용

struct namect {

char \* fname; // using pointers instead of arrays

char \* lname;

int letters;

};

void getinfo (struct namect \* pst)

{

char temp[SLEN];

printf("Please enter your first name.\n");

s\_gets(temp, SLEN);

// allocate memory to hold name

pst->fname = (char \*) malloc(strlen(temp) + 1);

// copy name to allocated memory

strcpy(pst->fname, temp);

printf("Please enter your last name.\n");

s\_gets(temp, SLEN);

pst->lname = (char \*) malloc(strlen(temp) + 1);

strcpy(pst->lname, temp);

}

#### 상기 두개의 strings은 structure 내부에 저장되지 않는다.

##### 두개 string은 malloc()에 의해 관리된 memory chunk에 저장된다

#### malloc()로 memory 할당을 했으면 free()으로 반환해야 한다

// Listing 14.10 The names3.c Program

// names3.c -- use pointers and malloc()

// malloc()을 사용하여 문제 해결하는 경우

#include <stdio.h>

#include <string.h> // for strcpy(), strlen()

#include <stdlib.h> // for malloc(), free()

#define SLEN 81

struct namect {

char\* fname; // using pointers

char\* lname;

int letters;

};

void getinfo(struct namect\*); // allocates memory

void makeinfo(struct namect\*);

void showinfo(const struct namect\*);

void cleanup(struct namect\*); // free memory when done

char\* s\_gets(char\* st, int n);

int main(void)

{

struct namect person;

struct names {

char first[10];

char last[10];

};

struct pnames {

char\* first;

char\* last;

};

struct names veep = { "Hong", "Bonghee" };

struct pnames treas = { "Boby", "Jones" };

printf("main::veep = %s and %s\n", veep.first, treas.first);

struct names accountant;

strcpy(accountant.first, "kim");

struct pnames attorney;

attorney.first = "HongGilDong";

attorney.last = "taxi";

printf("main::accountant = %s and %s\n", accountant.first, attorney.first);

puts("Enter the last name of your accountant:");

//scanf("%s", attorney.last);//예외 오류가 발생함

puts("Enter the last name of your attorney:");

//attorney = veep;

//attorney = treas;

printf("main::attorney.last = %s and attorney.first = %s\n", attorney.last, attorney.first);

scanf("%s", attorney.last);

getinfo(&person);

makeinfo(&person);

showinfo(&person);

cleanup(&person);

system("pause");

return 0;

}

void getinfo(struct namect\* pst)

{

char temp[SLEN];

printf("Please enter your first name.\n");

s\_gets(temp, SLEN);

// allocate memory to hold name

pst->fname = (char\*)malloc(strlen(temp) + 1);

//pst->fname = "Hong";

// copy name to allocated memory

strcpy(pst->fname, temp);

printf("Please enter your last name.\n");

s\_gets(temp, SLEN);

pst->lname = (char\*)malloc(strlen(temp) + 1);

strcpy(pst->lname, temp);

}

void makeinfo(struct namect\* pst)

{

pst->letters = strlen(pst->fname) +

strlen(pst->lname);

}

void showinfo(const struct namect\* pst)

{

printf("%s %s, your name contains %d letters.\n",

pst->fname, pst->lname, pst->letters);

}

void cleanup(struct namect\* pst)

{

free(pst->fname);

free(pst->lname);

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

char\* find;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

find = strchr(st, '\n'); // look for newline

if (find) // if the address is not NULL,

\*find = '\0'; // place a null character there

else

while (getchar() != '\n')

continue; // dispose of rest of line

}

return ret\_val;

}

## 14.7.8 Array of Structures를 사용한 functions

### function에서 array of structures를 처리하는 경우

#### array 이름은 address이다

### function에서 structure template를 접근할 수 있어야 한다

// Listing 14.13 The funds4.c Program

/\* funds4.c -- passing an array of structures to a function \*/

//coding 시험 대상임

#include <stdio.h>

#include <stdlib.h>

#define FUNDLEN 50

#define N 2

struct funds {

char bank[FUNDLEN];

double bankfund;

char save[FUNDLEN];

double savefund;

};

double sum(const struct funds money[], int n);

int main(void)

{

struct funds jones[N] = {

{

"Garlic-Melon Bank",

4032.27,

"Lucky's Savings and Loan",

8543.94

},

{

"Honest Jack's Bank",

3620.88,

"Party Time Savings",

3802.91

}

};

printf("The Joneses have a total of $%.2f.\n",

sum(jones, N));

system("pause");

return 0;

}

double sum(const struct funds money[], int n)

{

double total;

int i;

for (i = 0, total = 0; i < n; i++)

total += money[i].bankfund + money[i].savefund;

return(total);

}

### array name을 사용하여 function에 address of the first structure in the array을 전달.

#### 호출된 함수 내에서 array의 successive structures를 접근하기 위하여 array bracket notation을 사용한다

sum(&jones[0], N)

money = &jones[0];

##### 변수 money는 array jones[]의 첫번째 element를 point한다

##### money[0]은 array jones[]의 첫번째 element에 대한 또 다른 이름이다

#### sum() function은 original data를 변경할 목적이 아니므로, sum() function은 C const qualifier를 사용한다..

struct funds max(const struct funds m[]) { }

//사용 사례 실습

## 14.8 Unions: A Quick Look

### *union*은 same memory space에 다른 data type를 저장할 수 있게 하는 type이다 (but not simultaneously).

union hold {

int digit;

double bigfl;

char letter;

};

#### compiler는 union member 중에서 가장 큰 size를 갖는 변수 크기를 저장할 수 있는 공간을 할당한다

### array of unions를 사용

union hold fit; // union variable of hold type

union hold save[10]; // array of 10 union variables

union hold \* pu; // pointer to a variable of hold type

### union의 초기화.

#### union을 같은 type의 다른 union으로 초기화할 수 있다

#### union의 첫번째 element를 초기화할 수 있다

union hold valA;

valA.letter = 'R';

union hold valB = valA; // initialize one union to another

union hold valC = {88}; // initialize digit member of union

## 14.8.1 Unions 사용

### dot operator는 어떤 data type이 사용되는지를 알려준다

fit.digit = 23; // 23 is stored in fit; 2 bytes used

fit.bigfl = 2.0; // 23 cleared, 2.0 stored; 8 bytes used

fit.letter = 'h'; // 2.0 cleared, h stored; 1 byte used

#### ->operator를 사용하여 union에 대한 pointers를 사용할 수 있다

##### pointers to structures에 대한 pointer를 사용하는 것과 같다

pu = &fit;

x = pu->digit; // same as x = fit.digit

## 14.9 Enumerated Types

### *enumerated type*를 사용하여 symbolic names이 integer constants를 표현할 수 있다.

enum spectrum {red, orange, yellow, green, blue, violet};

enum spectrum color;

### enumerated types의 목적은 프로그램의 readability를 향상시키기 위함이다

int c;

color = blue;

if (color == yellow)

...;

for (color = red; color <= violet; color++) //compiler에 따라 지원이 안될 수 있다

...;

//listing14.15

/\* enum.c -- uses enumerated values \*/

#include <stdio.h>

#include <string.h> // for strcmp(), strchr()

#include <stdbool.h> // C99 feature

char\* s\_gets(char\* st, int n);

enum spectrum { red, orange, yellow, green, blue, violet };

const char\* colors[] = { "red", "orange", "yellow",

"green", "blue", "violet" };

#define LEN 30

int main(void)

{

char choice[LEN];

enum spectrum color;

bool color\_is\_found = false;

puts("Enter a color (empty line to quit):");

while (s\_gets(choice, LEN) != NULL && choice[0] != '\0')

{

for (color = red; color <= violet; color++)

{

if (strcmp(choice, colors[color]) == 0)

{

color\_is\_found = true;

break;

}

}

if (color\_is\_found)

switch (color)

{

case red: puts("Roses are red.");

break;

case orange: puts("Poppies are orange.");

break;

case yellow: puts("Sunflowers are yellow.");

break;

case green: puts("Grass is green.");

break;

case blue: puts("Bluebells are blue.");

break;

case violet: puts("Violets are violet.");

break;

}

else

printf("I don't know about the color %s.\n", choice);

color\_is\_found = false;

puts("Next color, please (empty line to quit):");

}

puts("Goodbye!");

return 0;

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

char\* find;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

find = strchr(st, '\n'); // look for newline

if (find) // if the address is not NULL,

\*find = '\0'; // place a null character there

else

while (getchar() != '\n')

continue; // dispose of rest of line

}

return ret\_val;

}

## 14.9.1 Shared Namespaces

### C는 *namespace*라는 용어를 사용하여 name이 인식되는 프로그램의 parts를 식별한다

#### same name을 사용하는 두개의 variables은 다른 scope에 있으면 충돌되지 않는다고 한다.

##### same name을 가진 두개의 변수가 same scope에 있으면 conflict라고 한다.

**#include <stdio.h>**

**namespace N {int a; int b;}**

**namespace M {int a; int c;}**

**using namespace N;**

**int main() {**

**char \*name = "hong";**

**int m = N::a;**

**//\*name = "a";**

**printf("%s, %p, %c\n", "we","we", \*"we");**

**printf("%s, %p, %c\n", "we","we", \*"we");**

**getchar();**

**return 0;**

**}**

#### Structure tags, union tags, enumeration tags는 같은 namespace에서 같이 사용될 수 없다

struct tag {

int a;

int b;

};

union tag {

int a;

float b;

};

struct tag x;

union tag y; //error

int tag; //허용됨

#### struct tag와 변수 tag는 충돌 없음

struct rect { double x; double y; };

int rect; // not a conflict in C

## 14.10 typedef : A Quick Look

### typedef은 type 이름을 사용자가 지정할 수 있게 하는 advanced data feature이다

#### #define과는 달리 typedef은 오직 type에 대한 symbolic names을 부여하는 용도로만 사용한다

* + - values을 정의하는 symbolic name으로 사용하지 않는다

#### ■ typedef 해석은 compiler가 하는 것이지 preprocessor가 하는 것이 아니다

##### typedef이 #define 보다 더 유연하다

typedef unsigned char BYTE;

BYTE x, y[10], \* z;

typedef struct {double x; double y;} rect;

rect r1 = {3.0, 6.0};

rect r2;

struct {double x; double y;} r1= {3.0, 6.0};

struct {double x; double y;} r2;

r2 = r1;

## 14.11 Fancy Declarations

int board[8][8]; // an array of arrays of int

int \*\* ptr; // a pointer to a pointer to int

int \* risks[10]; // a 10-element array of pointers to int

int (\* rusks)[10]; // a pointer to an array of 10 ints

int \* oof[3][4]; // a 3 x 4 array of pointers to int

int (\* uuf)[3][4]; // a pointer to a 3 x 4 array of ints

int (\* uof[3])[4]; // a 3-element array of pointers to 4-element arrays of int

### modifiers를 적용하는 순서를 이해하는 것이 필요하다

#### []: array 표현, (): function 표현 – 같은 precedence은 갖는다

* + 1. **[], () precedence은 \* indirection operator보다 더 높다**
    2. int \* risks[10];

#### [], (): associate from left to right.

* + 1. array of 12 arrays of 50 int s,
       1. not an array of 50 arrays of 12 int s:
    2. int goods[12][50];

#### [], ()은 같은 precedence를 갖는다 그리고 associate from left to right

* 1. ()를 먼저 적용한 후에 [] 적용

int (\* rusks)[10];

makes rusks a pointer to an array of 10 int s:

int \* oof[3][4];

* [3] has higher precedence than the \*
* [3]이 [4]보다 먼저 적용
* oof is a three-element array of four-element arrays of pointers to int , or, for short, a 3×4 array of pointers to int .

int (\* uuf)[3][4];

char \* fump(int); // **function** returning pointer to char

char (\* frump)(int); // **pointer to a function** that returns type char

char (\* flump[3])(int);// **array of 3 pointers to functions** that return type char

typdef int arr5[5];

typedef arr5 \* p\_arr5;

typedef p\_arr5 arrp10[10];

arr5 togs; // togs an array of 5 int

p\_arr5 p2; // p2 a pointer to an array of 5 int

arrp10 ap; // ap an array of 10 pointers to array-of-5-int

## 14.12 Functions and Pointers

### pointers가 functions을 가리키는 것이 가능하다

### function pointer가 또 다른 function의 argument로서 사용될 수 있다

#### qsort() function은 integer, string, 또는 structure를 sort하는 함수를 argument로 전달할 수 있다.

##### pointer to a function은 function code의 시작 부분을 가리키는 address를 가리킨다

##### function type을 지정하기 위하여 function signature를 사용한다

void ToUpper(char \*); // convert string to uppercase

### pointer, pf가 이 function type을 가리킨다:

void (\*pf)(char \*); // pf a pointer-to-function

### pf is a pointer to a function.

#### (\*pf) function은 (char \*)을 formal parameter로서 사용하며 return type은 void이다

##### function name이 pointer에 치환된다

void ToUpper(char \*);

void ToLower(char \*);

int round(double);

void (\*pf)(char \*);

pf = ToUpper; // valid, ToUpper is address of the function

pf = ToLower; // valid, ToLower is address of the function

pf = round; // invalid, round is the wrong type of function

pf = ToLower(); // invalid, ToLower() is not an address

void ToUpper(char \*);

void ToLower(char \*);

void (\*pf)(char \*);

char mis[] = "Nina Metier";

pf = ToUpper;

(\*pf)(mis); // apply ToUpper to mis (syntax 1)

pf = ToLower;

pf(mis); // apply ToLower to mis (syntax 2)

// Listing 14.16 The func\_ptr.c Program

// func\_ptr.c -- uses function pointers

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#define LEN 81

char\* s\_gets(char\* st, int n);

char showmenu(void);

void eatline(void); // read through end of line

void show(void(\*fp)(char\*), char\* str);

void ToUpper(char\*); // convert string to uppercase

void ToLower(char\*); // convert string to uppercase

void Transpose(char\*); // transpose cases

void Dummy(char\*); // leave string unaltered

int main(void)

{

char line[LEN];

char copy[LEN];

char choice;

void(\*pfun)(char\*); // points a function having a

// char \* argument and no

// return value

pfun = Dummy;

puts("Enter a string (empty line to quit):");

while (s\_gets(line, LEN) != NULL && line[0] != '\0')

{

while ((choice = showmenu()) != 'n')

{

switch (choice) // switch sets pointer

{

case 'u': pfun = ToUpper; break;

case 'l': pfun = ToLower; break;

case 't': pfun = Transpose; break;

case 'o': pfun = Dummy; break;

}

strcpy(copy, line);// make copy for show()

show(pfun, copy); // use selected function

}

puts("Enter a string (empty line to quit):");

}

puts("Bye!");

system("pause");

return 0;

}

char showmenu(void)

{

char ans;

puts("Enter menu choice:");

puts("u) uppercase l) lowercase");

puts("t) transposed case o) original case");

puts("n) next string");

ans = getchar(); // get response

ans = tolower(ans); // convert to lowercase

eatline(); // dispose of rest of line

while (strchr("ulton", ans) == NULL)

{

puts("Please enter a u, l, t, o, or n:");

ans = tolower(getchar());

eatline();

}

return ans;

}

void eatline(void)

{

while (getchar() != '\n')

continue;

}

void ToUpper(char\* str)

{

while (\*str)

{

\*str = toupper(\*str);

str++;

}

}

void ToLower(char\* str)

{

while (\*str)

{

\*str = tolower(\*str);

str++;

}

}

void Transpose(char\* str)

{

while (\*str)

{

if (islower(\*str))

\*str = toupper(\*str);

else if (isupper(\*str))

\*str = tolower(\*str);

str++;

}

}

void Dummy(char\* str)

{

// leaves string unchanged

}

void show(void(\*fp)(char\*), char\* str)

{

(\*fp)(str); // apply chosen function to str

puts(str); // display result

}

char\* s\_gets(char\* st, int n)

{

char\* ret\_val;

char\* find;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

find = strchr(st, '\n'); // look for newline

if (find) // if the address is not NULL,

\*find = '\0'; // place a null character there

else

while (getchar() != '\n')

continue; // dispose of rest of line

}

return ret\_val;

}