

| Basis and Practice in Programming

week 11

Structure

- User defined data structure
 - Consists of one or more basic data type

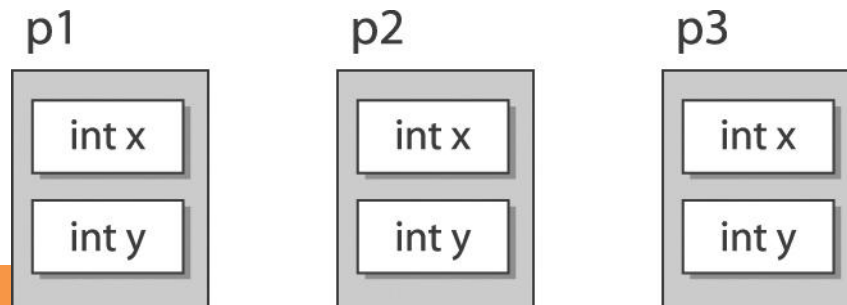
```
struct point           // declaring a structure named as 'point'
{
    int x;             // a member of the structure int x
    int y;             // a member of the structure int y
};
```

Structure

- Declaration of structure
 - Case 1

```
struct point {  
    int x;  
    int y;  
} p1, p2, p3;
```

```
int main(void)  
{  
    .....  
}
```



Structure

- Declaration of structure
 - Case 2

```
struct point {  
    int x;  
    int y;  
};  
  
int main(void)  
{  
    struct point p1, p2, p3;  
    ...  
    return 0;  
}
```

Structure

- Accessing structure variable

```
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point p1;
    p1.x=10;           // assign 10 to x, a member of p1
    p1.y=20;           // assign 20 to y, a member of p1
    ...
    return 0;
}
```

Structure

```
/* Week11 example 1 */

#include <stdio.h>
#include <math.h>

struct point{
    int x;
    int y;
};

int main(void)
{
    struct point p1, p2;
    double distance;

    fputs("Input x, y position of 1st point : ", stdout);
    scanf("%d %d", &p1.x, &p1.y);
```

```
    fputs(" Input x, y position of 1st point : ", stdout);
    scanf("%d %d", &p2.x, &p2.y);

    /* getting distance between the two points */
    distance = sqrt((p1.x-p2.x)*(p1.x-p2.x)
        +p1.y-p2.y)*(p1.y-p2.y));

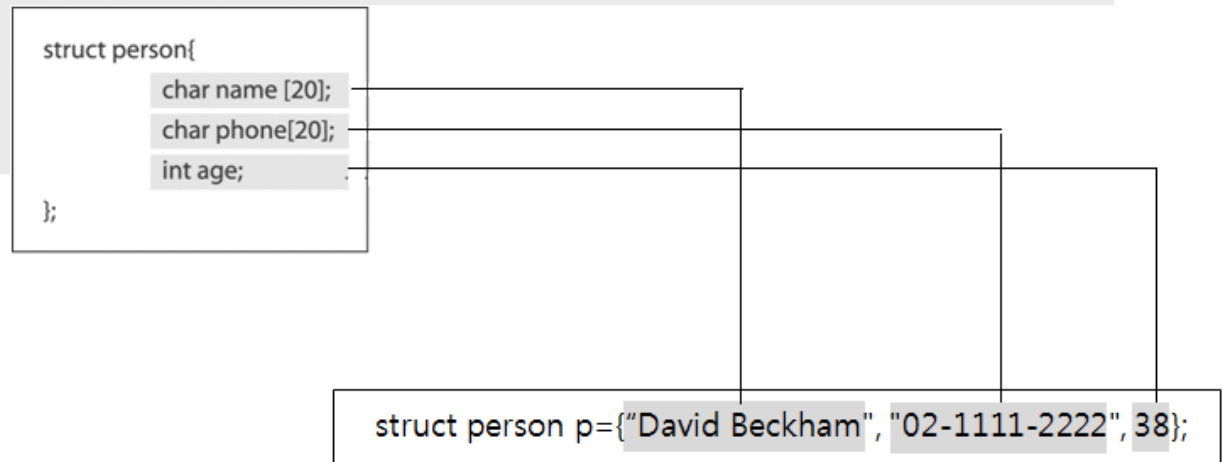
    printf("Distance of two points : %f\n", distance);

    return 0;
}
```

Structure

- Initialization of structure variable

```
struct person {  
    char name[20];  
    char phone[20];  
    int age;  
};  
int main (void)  
{  
    struct person p={"David Beckham", "02-1111-2222", 38};  
    ...  
    return 0;  
}
```



Structure

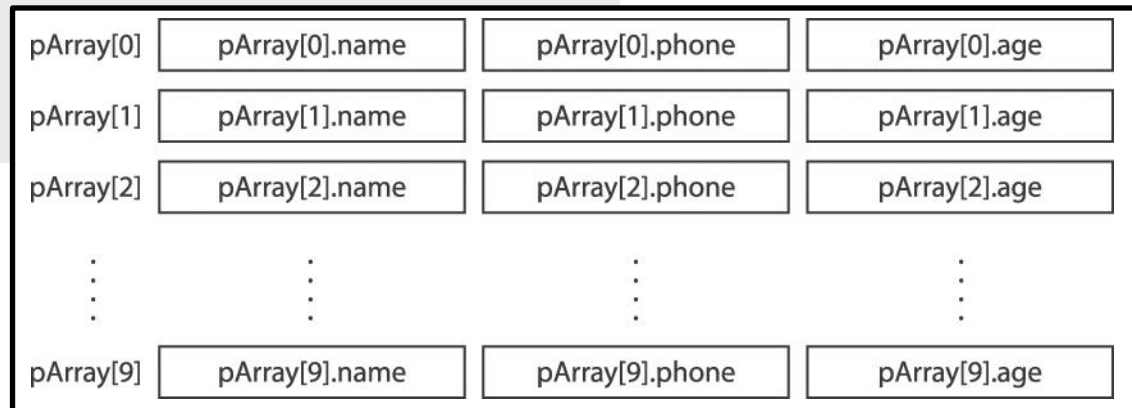
```
/* Week11 example 2 */  
  
#include <stdio.h>  
#include <string.h>  
  
struct person  
{  
    char name[20];  
    char phone[20];  
};  
  
int main(void)  
{  
    struct person p;  
  
    strcpy(p.name, "Daniel Henny");  
    strcpy(p.phone, "02-1212-2323");  
  
    printf("name : %s, phone : %s\n", p.name, p.phone);  
  
    return 0;  
}
```


Structure

- Declaration of array of structure

```
struct person {
    char name[20];
    char phone[20];
    int age;
};

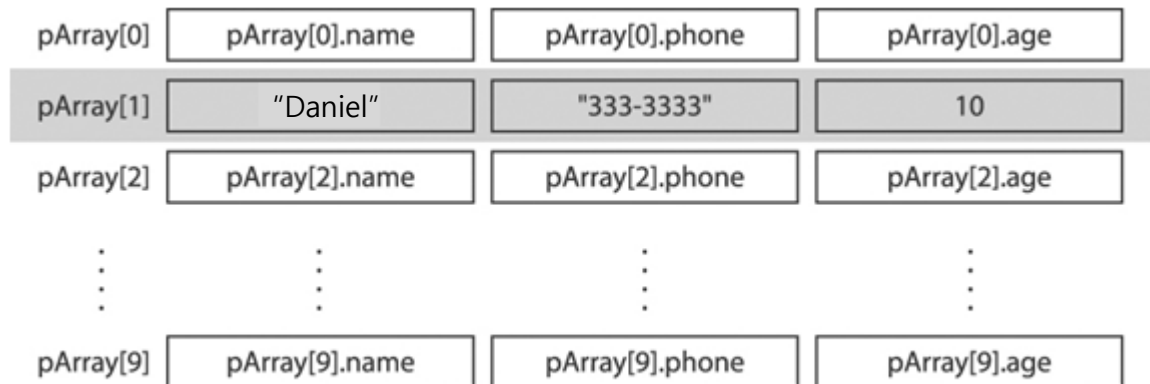
int main (void)
{
    struct person pArray[10];
    ...
    return 0;
}
```



Structure

- Accessing an element of structure array

```
pArray[1].age=10;           // accessing age of 2nd element  
strcpy(pArray[1].name, "Daniel"); // accessing name of 2nd element  
strcpy(pArray[1].phone, "333-3333"); // accessing phone of 2nd element
```



Structure

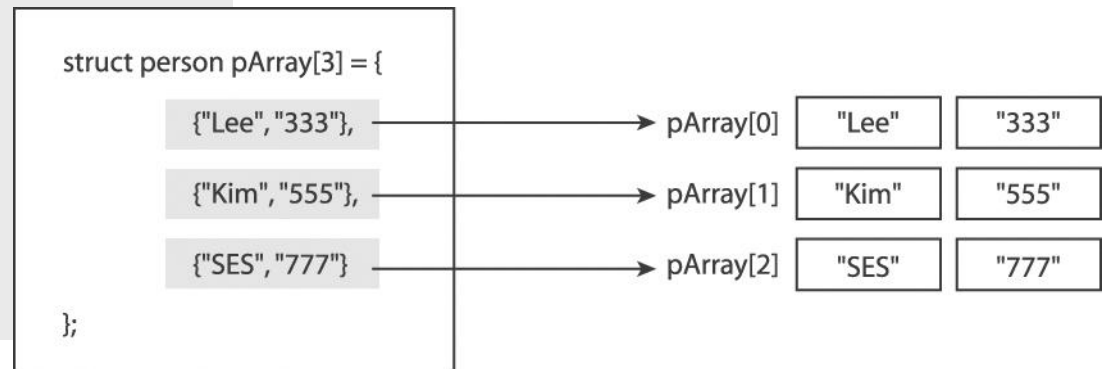
```
/* Week11 example 3 */  
  
#include <stdio.h>  
  
struct person {  
    char name[20];  
    char phone[20];  
};  
  
int main (void)  
{  
    struct person pArray[3];  
    int i;  
  
    for(i=0; i<3; i++) // Input data  
    {  
        printf("Input name and phone number : ");  
        scanf("%s %s", pArray[i].name, pArray[i].phone);  
    }  
}
```

```
printf("\nResult of input information\n");  
  
for(i=0; i<3; i++) //Output data  
{  
    printf("Name : %s, ", pArray[i].name);  
    printf("Phone : %s\n", pArray[i].phone);  
}  
  
return 0;  
}
```

Structure

- Initialization of structure array

```
struct person {  
    char name[20];  
    char phone[20];  
};  
  
int main (void)  
{  
    struct person pArray[3]={  
        {"Lee", "333"},  
        {"Kim", "555"},  
        {"SES", "777"}  
    };  
    .....  
    return 0;  
}
```



Structure

- Structure and pointer
 - Case 1
 - Declares structure pointer and references structure variable
 - Case 2
 - Pointer variable is declared as a member of structure

Structure

```
/* Week11 example 4 */

#include <stdio.h>

struct person {
    char name[20];
    char phone[20];
};

int main()
{
    struct person man={"Thomas", "354-00xx"};
    struct person * pMan;
    pMan=&man;

    // using structure variable
    printf("name : %s\n", man.name);
    printf("phone : %s\n", man.phone);
```

```
    // using structure pointer 1
    printf("name : %s\n", (*pMan).name);
    printf("phone : %s\n", (*pMan).phone);

    // using structure pointer 2
    printf("name : %s\n", pMan->name);
    printf("phone : %s\n", pMan->phone);

    return 0;
}
```

Structure

```
/* Week11 example 5 */
```

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

```
struct perInfo {  
    char addr[30];  
    char tel[20];  
};
```

```
struct person {  
    char name[20];  
    char pID[20];  
    struct perInfo* info;  
};
```

```
int main()
```

```
{  
    struct perInfo info={"Korea Seoul", "333-4444"};  
    struct person man={"Mr. Lee", "820204-xxxx512"};  
  
    man.info=&info;  
  
    printf("name : %s\n", man.name);  
    printf("pID : %s\n", man.pID);  
    printf("addr : %s\n", man.info->addr);  
    printf("tel : %s\n", man.info->tel);  
  
    return 0;  
}
```

Structure

```
/* Week11 example 6 */
#include <stdio.h>

struct person {
    char name[20];
    char pID[20];
    struct person* frnd;
};

int main()
{
    struct person man1={"Mr. Lee", "820204-0000512"};
    struct person man2={"Mr. Lee's Friend", "820000-0000101"};

    man1.frnd=&man2;
    printf("[Mr. Lee]\n");
    printf("name : %s\n", man1.name);
    printf("pID  : %s\n", man1.pID);

    printf("[His Friend]\n");
    printf("name : %s\n", man1.frnd->name);
    printf("pID  : %s\n", man1.frnd->pID);

    return 0;
}
```


Structure

- Structure variables as a function argument
 - The same as passing basic data type variables as function arguments
- Operations of structure variable
 - The only permitted operation is 'assign' (=)
 - The four fundamental arithmetic operations are not granted

Structure

```
/* Week11 example 7 */
#include <stdio.h>

struct simple {
    int data1;
    int data2;
};

void show(struct simple ts);

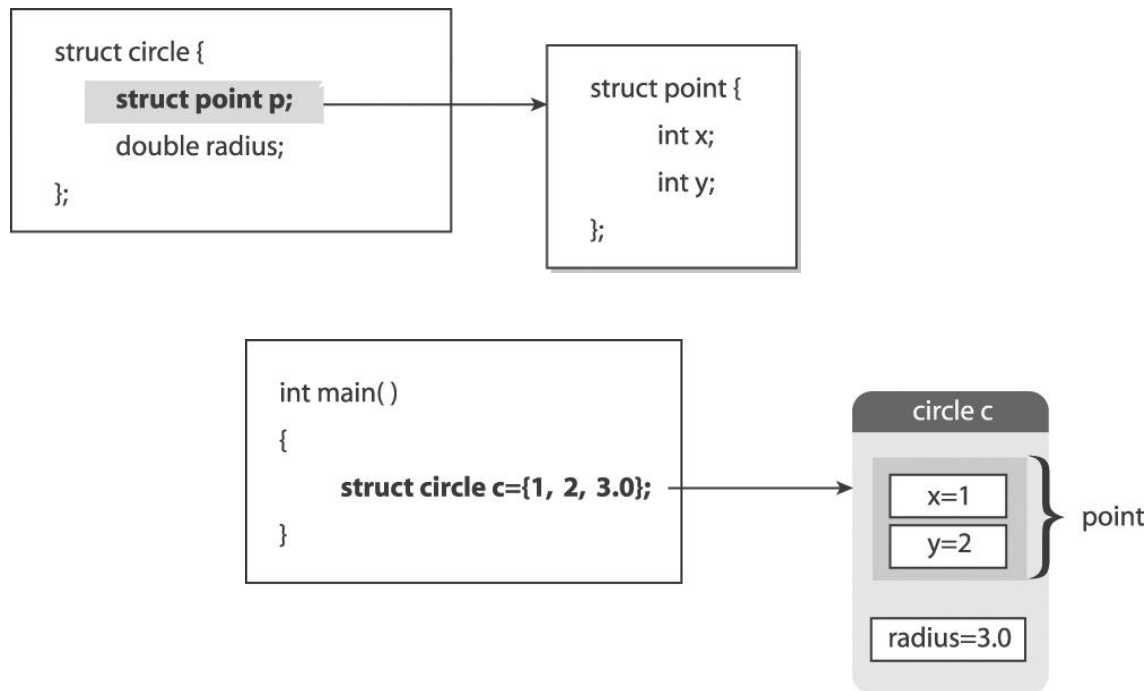
int main()
{
    struct simple s1={1, 2};
    struct simple s2;

    s2=s1;
    show(s2);
    return 0;
}

void show(struct simple ts)
{
    printf("data1:%d, data2:%d\n", ts.data1, ts.data2);
}
```

Structure

- Overlapped structure
 - A structure has a structure as its own member
 - A structure in a structure



Structure

```
/* Week11 example 8 */
```

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

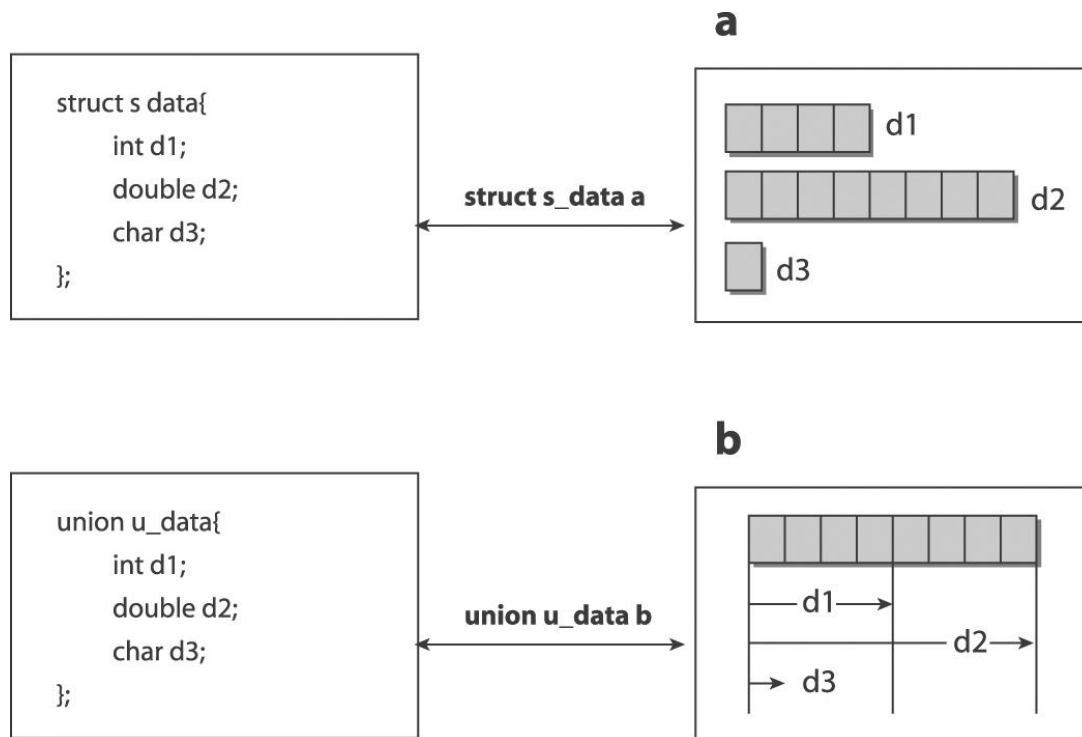
```
struct point{  
    int x;  
    int y;  
};
```

```
struct circle {  
    struct point p;  
    double radius;  
};
```

```
int main()  
{  
    struct circle c1={10, 10, 1.5};  
    struct circle c2={{30, 30}, 2.4};  
  
    printf("[circle1] \n");  
    printf("x:%d, y:%d \n", c1.p.x, c1.p.y);  
    printf("radius:%f \n",c1.radius);  
  
    printf("[circle2] \n");  
    printf("x:%d, y:%d \n", c2.p.x, c2.p.y);  
    printf("radius:%f \n",c2.radius);  
  
    return 0;  
}
```

Structure

- Union
 - Several variables share the same memory space



Structure

```
/* Week11 example 9 */
#include <stdio.h>

union u_data{
    int d1;
    double d2;
    char d3;
};

int main (void)
{
    union u_data data;

    data.d2=3.3;
    printf("%d, %f, %c \n", data.d1, data.d2, data.d3);

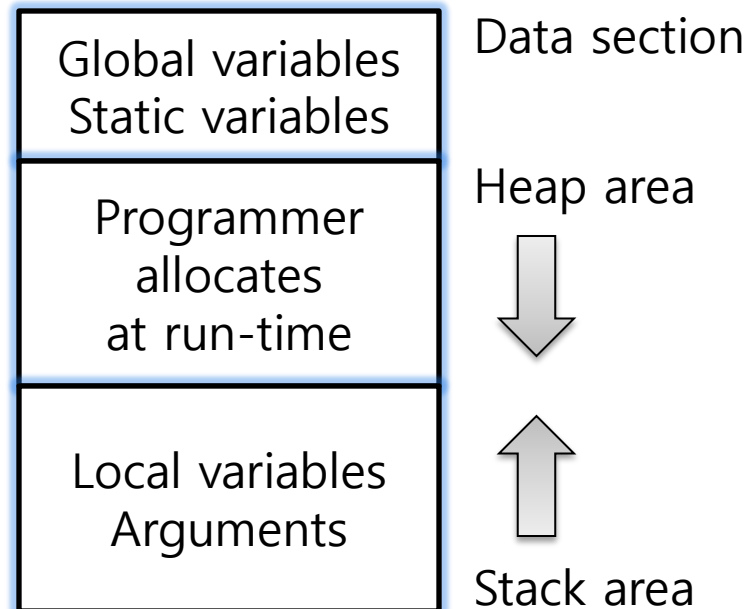
    data.d1=2;
    printf("%d, %f, %c \n", data.d1, data.d2, data.d3);

    data.d3='a';
    printf("%d, %f, %c \n", data.d1, data.d2, data.d3);

    return 0;
}
```

Structure

- Stack, heap and data section
 - Memory space allocated to run program
 - The stack size how much functions request must be determined at compile time



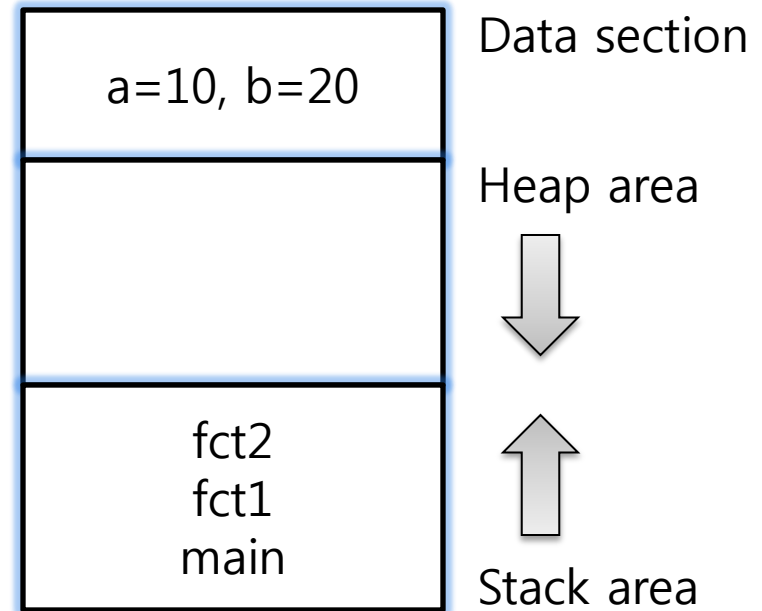
Structure

```
void fct1(int);
void fct2(int);

int a=10;
int b=20;

int main (void)
{
    int m=123;

    fct1(m);
    fct2(m);
    return 0;
}
void fct1(int c){
    int d=30;
}
void fct2(int e){
    int f=40;
}
```



Structure

- Dynamic memory allocation
 - Allocating memory space with determined size at run-time (on heap area)

```
#include <stdlib.h>
```

```
void* malloc(size_t size)
```

if successful, it returns the address of allocated memory space

Otherwise, returns NULL pointer

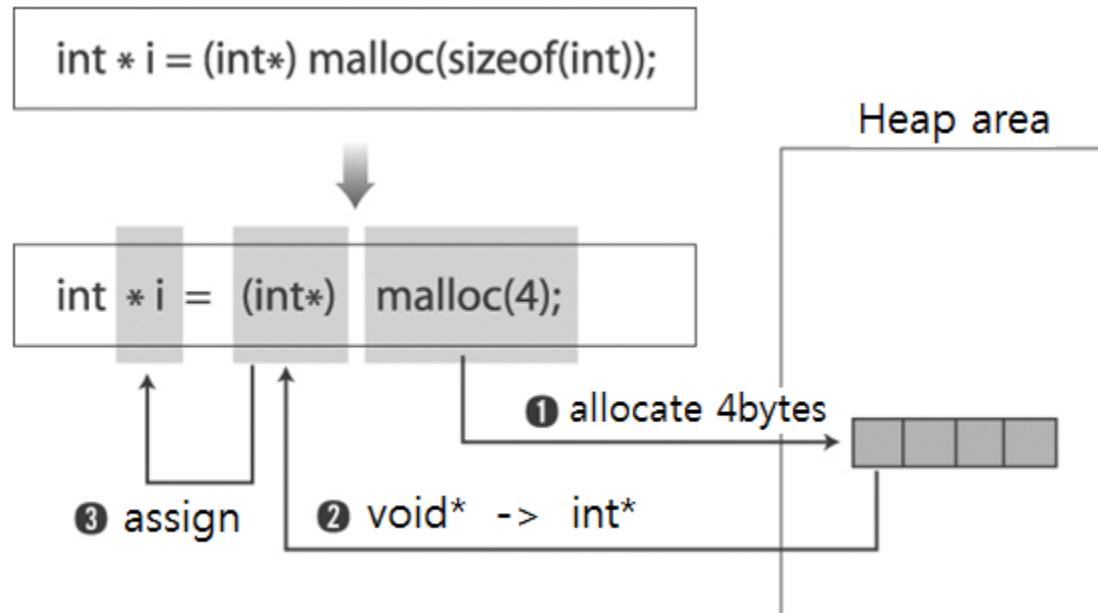
- Freeing memory space dynamically allocated

```
#include <stdlib.h>
```

```
void free(void* ptr)
```

Structure

- Malloc function



Structure

```
/* Week11 example 10 */  
  
#include <stdio.h>  
#include <stdlib.h>  
  
int main (void)  
{  
    int* pi;  
  
    pi = (int*)malloc(sizeof(int));  
    *pi = 3;  
  
    printf("%d\n", *pi);  
  
    free(pi);  
  
    return 0;  
}
```

Structure

- Exercise
 - Write an address book program
 - Define a structure named 'Contact' including three members of name(char[20]), age(int), phone number(char[15])
 - Make 10 elements pointer array referencing the Contact structure
 - Allocate memory space dynamically by size of the Contact structure using malloc function
 - Input the information of 10 persons from user and save it to the newly allocated memory space through structure pointer
 - Then print out all information of 10 persons