자료구조 보고서

Homework#2

학과 : 소프트웨어학과
학번 : 2018038051
이름 : 정종현

LAB2-1

(a) 내용

Windows+GNU GCC + VSCode 환경을 선택하여 과제를 진행하였습니다.

lab2-1은 char, int, float, double형 타입에 대한 각 각의 변수의 크기를 구해보았고, 자료형에 대한 변수의 크기를 비교해보았습니다. 비교 결과 서로 다르지 않음을 알 수 있었습니다. 그러나 각 자료형에 대한 포인터 변수의 크기는 자료형에 상관없이 PC의 운영체제 혹은 PC가 사용하고 있는 컴파일러의 비트 수에 영향을 받음을 알 수 있었습니다. 본인의 PC는 64bit형 운영체제를 사용 중이라 포인터 변수의 크기를 8byte를 예상하였으나, 확인 결과 컴파일러의 비트수에 영향을 받는다는 것을 알 수 있었고, VSCode의 우측 하단에서 Win32를 확인 할 수 있었습니다.

(b) 코드

lab2-1.c

```
#include <stdio.h>
int main(){
    printf("---------[정종현] [2018038051]----------\n\n");
    char charType;
    int integerType;
    float floatType;
    double doubleType;
    printf("Size of char : %ld byte\n",sizeof(charType));//char형 타입의 변수의 크기 : 1byte
    printf("Size of int : %ld byte\n",sizeof(integerType));//int형 타입의 변수의 크기 : 4byte
    printf("Size of float : %ld byte\n",sizeof(floatType));//float형 타입의 변수의 크기 : 4byte
    printf("Size of double : %ld byte\n",sizeof(doubleType));//double형 타입의 변수의 크기 : 8byte
    printf("==========================================\n");
    printf("Size of char: %ld byte\n",sizeof(char));//char형 타입의 크기 : 1byte
    printf("Size of int: %ld byte\n",sizeof(int));//int형 타입의 크기 : 4byte
    printf("Size of float: %ld byte\n",sizeof(float));//float형 타입의 크기 : 4byte
    printf("Size of double: %ld byte\n",sizeof(double));//double형 타입의 크기 : 8byte
printf("==========================================\n");
    printf("Size of char* : %ld byte\n",sizeof(char*));//포인터변수 크기는 운용체제에 따라 정해진 크기를 갖는다.
    printf("Size of int* : %ld byte\n",sizeof(int*));//64bit 운영체제의 pc를 사용 중이라 8byte가 나올 것을 예상
    printf("Size of float* : %ld byte\n",sizeof(float*));//예상과는 다르게 4byte가 나와 알아본 결과 32bit로 컴파일을 하면 4byte가 나온다고 하였다.
    printf("Size of double* : %ld byte\n",sizeof(double*));
    printf("==========================================\n");
    return 0;
}
```

(c) 결과

lab2-1.c 결과

```
> Executing task in folder 객체지향프로그래밍: cmd /C 'C:\충북대학교\소프트웨어
학과\2학년\1학기 수업\자료구조\2주차\lab2-1' <

--------[정종현] [2018038051]---------

Size of char : 1 byte
Size of int : 4 byte
Size of float : 4 byte
Size of double : 8 byte
==========================================
Size of char: 1 byte
Size of int: 4 byte
Size of float: 4 byte
Size of double: 8 byte
==========================================
Size of char* : 4 byte
Size of int* : 4 byte
Size of float* : 4 byte
Size of double* : 4 byte
==========================================

터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

LAB2-2

(a) 내용

변수 i와 i를 가르키는 포인터 변수 ptr, 그리고 ptr을 가리키는 이중포인터 변수 dptr의 관계에 대한 내용을 다룬다. ptr 과 dptr이 i와 ptr을 참조하였을 때, ptr과 dptr의 값은 자신이 가리키고 있는 변수의 주소값이 들어갔음을 확인 할 수 있다. 또한 ptr과 dptr이 i의 변수의 값에 접근하기 위해서는 *ptr, **dptr과 같은 형식을 사용해야함을 알 수 있다.

(b) 코드

lab2-2.c

```
#include<stdio.h>
int main(){
    int i; int *ptr; int **dptr;
    i=1234;
    printf("--------[정종현] [2018038051]---------\n\n");
    printf("[checking values before ptr = &i]\n");//포인터 변수에 i의 주소값을 넣기 전 체크
    printf("value of i == %d\n",i); //i의 값 출력
    printf("address of i == %p\n",&i); //i의 주소값 출력
    printf("value of ptr == %p\n",ptr); //ptr의 값 출력 의미없는 값이 있다.
    printf("address of ptr == %p\n",&ptr);//ptr의 주소값 출력

    ptr=&i; /*ptr is now holding the address of i // ptr에 i의 주소값을 대입*/

    printf("\n[checking values after ptr=&i]\n");
    printf("value of i == %d\n",i);
    printf("address of i == %p\n",&i);
    printf("value of ptr == %p\n",ptr); //ptr이 i의 주소값을 가지게 됨.
    printf("address of ptr == %p\n",&ptr);
    printf("value of *ptr == %d\n",*ptr); //ptr의 역참조 값은 i의 값

    dptr=&ptr; /*dptr is now holding the address of ptr*/ //dptr 에 ptr의 주소값을 대입

    printf("\n[checking values after dptr=&ptr]\n");
    printf("value of i == %d\n",i);
    printf("address of i == %p\n",&i);
    printf("value of ptr == %p\n",ptr);
    printf("address of ptr == %p\n",&ptr);
    printf("value of *ptr == %d\n",*ptr);
    printf("value of dprt == %p\n",dptr); //dptr의 값은 ptr을 가르키므로 ptr의 주소값을 가짐
    printf("address of dptr == %p\n",&dptr);
    printf("value of *dptr == %p\n",*dptr);//dptr의 역참조값은 ptr이 가르키는 i의 주소값임.
    printf("value of **dptr == %d\n",**dptr);//**dptr은 i의 값

    *ptr=7777; /*changing the value of *ptr */

    printf("\n[after *ptr = 7777]\n");//*ptr은 i의 값을 의미하므로 i의 값이 바뀜.
    printf("value of i == %d\n",i);
    printf("value of *ptr == %d\n",*ptr);
    printf("value of **dptr == %d\n",**dptr);
    **dptr = 8888; /*changing the value of **dptr*/
    printf("\n[after **dptr = 8888]\n"); //**dptr은 i의 값을 의미하므로 i의 값이 8888이 된다.
    printf("value of i == %d\n",i);
    printf("value of *ptr == %d\n",*ptr);
    printf("value of **dptr == %d\n",**dptr);
    return 0;
}
```

(C) 결과

lab2-2.c 결과

```
> Executing task in folder 객체지향프로그래밍: cmd /C 'C:\충북대학교\소프트웨어학과\2학년\1학기 수업\자료구조\2
주차\lab2-2' <

--------[정종현] [2018038051]---------

[checking values before ptr = &i]
value of i == 1234
address of i == 0061FF1C
value of ptr == 0022C000
address of ptr == 0061FF18

[checking values after ptr=&i]
value of i == 1234
address of i == 0061FF1C
value of ptr == 0061FF1C
address of ptr == 0061FF18
value of *ptr == 1234

[checking values after dptr=&ptr]
value of i == 1234
address of i == 0061FF1C
value of ptr == 0061FF1C
address of ptr == 0061FF18
value of *ptr == 1234
value of dprt == 0061FF18
address of dptr == 0061FF14
value of *dptr == 0061FF1C
value of **dptr == 1234

[after *ptr = 7777]
value of i == 7777
value of *ptr == 7777
value of **dptr == 7777

[after **dptr = 8888]
value of i == 8888
value of *ptr == 8888
value of **dptr == 8888

터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

(D) Lab 2-2.c 메모리 레이아웃