TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

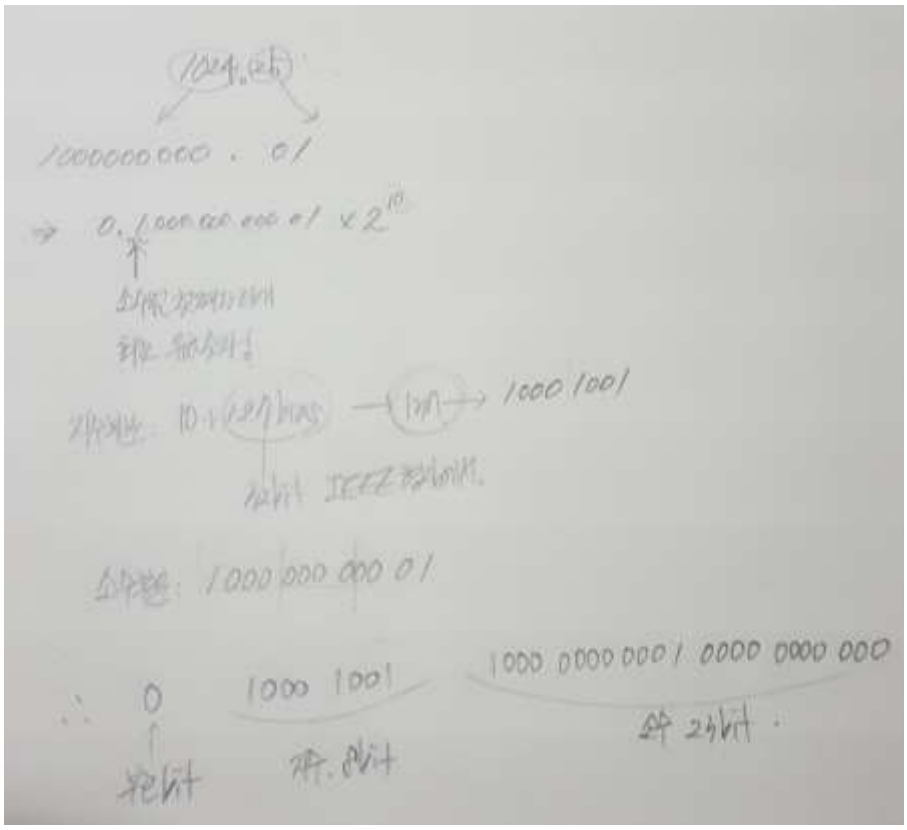
C 언어 모의시험 1

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 - 정유경
ucong@naver.com

C 언어 모의시험 1

1. float 자료형에 값 1024.25 를 줬다.
이값이 어떻게 bit 로 표현되는지 작성하시오.
(힌트 : 부호 - 1bit, 지수 - 8bit, 소수점 부분 - 23bit)



2. 변수의 정확한 정의를 기술하시오.
데이터를 저장하는 공간

3. 실수와 정수를 입력받고 그 값을 함수의 인자로 넘겨
두 수의 곱을 실수형으로 반환하여 출력하시오.

```c
#include <stdio.h>

double Multiply(double d, int i)
{
	return d * i;
}
int main(void)
{
	double d = 0, res = 0;
	int i = 0;

	scanf("%lf %d",&d,&i);

	printf("%lf * %d = %lf₩n", d, i, Multiply(d, i));
	return 0;
}
```

4. goto 문을 어떤 경우에 사용하면 효율적인지 작성하시오.
- 다중루프 빠져나올 때 (break 문은 한 수준의 루프만 종료하므로, 많이 중첩된 루프를 종료하려면 goto 문을 사용)
- 예외처리
- goto 문은 프로그램의 구조를 해치기 때문에 goto 문을 사용한 소스는 이식성과 재사용에 무척 불리하다.
그러나 시스템 프로그램이나 디바이스 드라이버같이 이식성이나 유지의 편의성, 가독성보다는 무엇보다 성능을 최우선으로 하는 곳에는 여전히 goto 가 사용된다. 왜냐하면 goto 는 컴퓨터가 알아들을 수 있는 유일한 제어문이며 또한 컴퓨터가 가장 쉽게 실행할 수 있는 제어문이기 때문이다. 컴퓨터는 for 나 while 같은 건 모른다. 오로지 정해진 번지로 점프하는 것만 가능하며 컴파일러가 for, while 을 기계가 알아들을 수 있는 점프문으로 바꿔주는 것이다.

5. C 언어에서 중요시 여기는 Memory 구조에 대해 기술하시오.
프로그램을 실행시키면 운영체제는 우리가 실행시킨 프로그램을 위해 메모리 공간을 할당해준다.

할당되는 메모리 공간은 크게 스택(Stack), 힙(Heap), 데이터(Data)영역으로 나뉘어진다.

데이터(Data) 영역 - 전역 변수와 static 변수가 할당되는 영역 , 프로그램의 시작과 동시에 할당되고, 프로그램이 종료되어야 메모리에서 소멸됨

스택(Stack) 영역 - 함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역, 함수 호출이 완료되면 사라짐

힙(Heap) 영역 - 필요에 의해 동적으로 메모리를 할당 할 때 사용
스택 영역에 할당될 메모리의 크기는 컴파일 타임(컴파일 하는 동안)에 결정
사용자의 요구에 맞게 메모리를 할당해 주기 위해서는(런타임에 메모리 크기를 결정하고 싶을 때) 메모리 동적 할당을 통해 힙 영역에 메모리를 할당해야 한다.

6. 다음 문제의 빈칸을 채우시오.
C 에서 함수를 호출하고 사용하기 위해서는 반드시 (스택)이 필요하기 때문에
재귀호출을 할 경우에 계속해서 (스택)을 생성한다.

7. 1 ~ 1000 사이에서 짝수와 홀수의 합을 각각 구하고
짝수의 합으로 홀수의 합을 나눈 결과를 정수형으로 출력하라.

```c
#include <stdio.h>
int main(void)
{
	int SumEven = 0, SumOdd = 0;
	int res = 0;
	for (int i = 0; i<1001; i++)
	{
		if (i % 2 == 0) // 짝수이면
			SumEven += i;
		else
			SumOdd += i;
	}
```

```
        printf("홀수의 합: %d, 짝수의 합: %d₩n", SumOdd, SumEven);
        printf("%d / %d = %d₩n", SumOdd, SumEven, SumOdd / SumEven);
        return 0;
}
```

8. 7 과 4 이라는 2 개의 정수를 입력받아 num1, num2 에 저장하였다.
num1 << num2, num1 >> num2 의 값과
num1 과 num2 의 and, or, xor bit 연산을 수행하는 프로그램을 작성하시오.

```
#include <stdio.h>

int main(void)
{

        int num1, num2;
        printf("숫자 2 개를 입력하세요₩n");
        scanf("%d %d", &num1, &num2);

        printf("1. %d << %d = %d₩n", num1, num2, num1 << num2);
        printf("2. %d >> %d = %d₩n", num1, num2, num1 >> num2);

        printf("3. %d 와 %d 의 AND 결과: %d₩n", num1, num2, num1&num2);
        printf("4. %d 와 %d 의 OR 결과: %d₩n", num1, num2, num1 | num2);
        printf("5. %d 와 %d 의 XOR 결과: %d₩n", num1, num2, num1^num2);

        return 0;
}
```

9. Debugging(디버깅)을 왜 해야하는지 이유를 기술하고
Linux 환경에서 terminal 에 어떤 명령어를 입력하여 컴파일하는지 기술하시오.

디버깅은 소스코드를 분석하여 메모리에 저장된 변수들의 값을 살펴보는 데에 꼭 필요하다.
문법적으로 오류가 없어서 성공적으로 컴파일은 되지만, 논리적 오류가 존재하는 경우 디버깅을 수행하여 오류를 찾아낸다.
Gcc -g filename.c newname.c → 컴파일 후 디버깅 파일 생성

cat [파일명] : 파일의 내용을 본다
rm [옵션] [파일명] : 파일(디렉토리) 삭제
mv [옵션] [파일명] : 파일의 위치를 바꾸거나 이름을 바꿀 때 사용. 덮어쓰기하므로 주의!
mv file1 file2: file1 의 이름을 file2 라고 바꾼다
pwd : 현재 작업 디렉토리를 절대 경로 명으로 출력한다.
ls [파일명] : 현재 디렉토리 내의 파일과 리스트 출력
cd [디렉토리명] : (change directory) 작업 디렉토리를 바꾼다.
cp [옵션] [원본파일명] [목적 파일명 / 디렉토리명]
하나 이상의 원본 파일을 목적 파일이나 디렉토리로 복사한다.
mkdir [옵션] [파일명]
설명 : 새로운 디렉토리를 만든다.

10. do while 문을 사용하는 이유에 대해 기술하시오.
조건에 관계없이 한번은 실행하기 위해

11. 표준 상태에서 아래의 소스 코드가 컴파일이 되는지 안되는지 판정하시오.
만약 컴파일이 안된다면 어떻게 바꿔야 하는지 기술하시오.

```
#include <stdio.h>
#define inc_each(x, y) { x++; y++; } //2줄이라서.....!!!
int main(void)
{
	int x = 10, y = 5;
	if (x > y)
	{
		inc_each(x, y); 중괄호로 묶어주어야 한다!!
	}
	else
		x = y;

	printf("%d, %d", x, y);
	return 0;
}
```

12. 다음 빈칸을 채우시오.
컴퓨터 구조상 가장 빠른 것은 (레지스터)이고 가장 느린것은(디스크)이다.
컴퓨터의 메모리 계층 구조(Memory hierarchy)는 속도가 빠른순으로 레지스터>캐시>메모리>하드디스크

13. 이중 for 문을 사용하여 구구단을 작성해보시오.

```
#include <stdio.h>
int main(void)
{
	int i, j;
	for (i = 0; i<9; i++)
	{
		printf("구구단 %d 단₩n", i + 1);
		for (j = 0; j<9; j++)
			printf("%d * %d = %d₩n", (i + 1), (j + 1), (i + 1)*(j + 1));
	}
}
```

14. 아래의 소스 코드를 보고 물음에 답하시오.
이 코드에 오류가 존재하는지 않는지를 판별하고
문제가 있다면 어디를 어떻게 수정하는 것이 좋을지 기술하시오.

```
#include <stdio.h>
int main(void)
{
	int number = 1;
	for (;;)
	{
		printf("number=%d₩n", number);
```

```
                number += number;
                if (number == 1000)
                        break;
        }
        return 0;
}
```

number 는 1000 이 될수 없다. 1,2,4,8,16,32,64... 2 의 거듭제곱으로 증가하기 때문
반복문 탈출조건을 바꾸어준다. (Number > 1000)

15. 현재 디버깅중이며 메모리 주소 0x7ffffb70 에 어떤 값이 저장되었다.
이 때 위 메모리에 저장된 값을 보고 싶다면 어떻게 해야 하는가 ?

```
(gdb) p main // main 함수가 존재하는 메모리 주소
(gdb) p I //변수의 내용
(gdb) p &I // 변수의 주소
(gdb) p *((char*)0x804834A) //메모리 주소의 내용 출력
(gdb) p i * 21 - 2
```

16. 다음 어셈블리어를 보고 함수의 main 함수부터 my_function 함수까지
stack 의 동작 방식을 그림과 함께 기술하라. **?!?!**

```
int my_function(int num1, int num2, int num3)
{
        return num2 * num3 - num1;
}

int main(void)
{
        int result = 0;
 int num1 = 2, num2 = 3, num3 = 4;
result = my_function(num1, num2, num3);
printf("result = %d₩n", result);
 return 0;
}
```

17. 배열에 아무 문자열이나 입력받아 각 문자가 홀수인 경우에만 출력하는 프로그램을 작성하라.

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

int main(void)
{
        char arr[30];
        printf("문자열 입력₩n");
        //scanf("%s",arr); //1. 공백을 포함하지 않는 문자열을 입력받는다

        gets(arr); //2. 공백을 포함하는 문자열을 입력받는다
        int size=strlen(arr); // 공백 포함하지 않는 문자열 길이 반환
        for(int i=0;i<size;i++)
        {
```

```
            if(*i%2==0) //i 가 0 부터 시작하므로 i 가 짝수번째일때가 문자열에서
는 홀수번째
                        printf("%c ", arr[i]);
        }



        return 0;
}
~
```

18. 다음 빈칸을 채우시오.
배열의 이름은 (주소)이고, 포인터는 (주소)을 저장하는 (변수)이다.

19. 구조체를 사용하는 이유에 대하여 기술하라.
변수를 여러 개 묶어서 사용하기 위해

20-22 번 복합 문제
20. 20 개의 배열에 rand()를 이용하여 무작위로 1 ~ 100 까지의 값을 저장하는 프로그램을 작성하라.
21. 그리고 각 배열의 요소의 홀수들과 짝수들의 합을 구해 출력하라.
22. 홀수들의 합과 짝수들의 합을 곱하여 출력하라.

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

int main(void)
{
        char arr[20]={0,};
        int SumOdd=0, SumEven=0;

        for(int i=0; i<20; i++)
        {
                arr[i]=(rand()%100)+1;//1~100 까지 무작위 난수생성
                if(arr[i]%2) // 홀수
                        SumOdd+=arr[i];
                else        //짝수
                        SumEven+=arr[i];
        }
        printf("1. 홀수의 합: %d 짝수의 합: %d₩n", SumOdd, SumEven);
        printf("2. 홀수의 합과 짝수의 합의 곱: %d₩n", SumOdd*SumEven);
        return 0;
}
```

23. 대문자를 입력하면 소문자가 나오도록
소문자를 입력하면 대문자가 나오도록 프로그램을 작성하시오.

방법 1

```
#include <stdio.h>

int main(void)
{
	char c;
	scanf("%c",&c);

	if(c>=97 && c<=122) // 97~122 의 소문자이면
		printf("%c₩n",c-32);
	else if(c>=65 && c<=90) // 65~90 의 대문자이면
		printf("%c₩n",c+32);

	return 0;
}
```

방법 2. 32(100000)랑 xor 하는 방법

```
#include <stdio.h>
int main(void)
{
	char ch;
	scanf("%c",&ch);
	printf("%c₩n",ch^32);
	return 0;
}
```

24-25 복합 문제

24. 어떤 정수에 값 13 이 들어 있다.

이것을 4 중 포인터를 사용하여 표현해보라.

25. 4 중 포인터를 사용하여 표현했다면

현재 값인 13 을 14 로 증가시켜보자.

(num++을 사용하지 말라)

```
#include <stdio.h>


int main()
{
	int num =13;
	int *p=&num;
	int **pp=&p;
	int ***ppp=&pp;
	int ****pppp=&ppp;
	printf("4 중 포인터로 표현한 p: %d₩n", ****pppp);
	printf("p 값을 하나 증가: %d₩n", (****pppp)+1);
```

```
	return 0;
}
```

26. 행렬의 곱셈을 프로그램으로 작성하라.

```
#include <stdio.h>

int main(void)
{

	int matA[2][2]={1,2,3,4};
	int matB[2][2]={5,6,7,8};
	int mat[2][2]={0,};



	mat[0][0]=matA[0][0]*matB[0][0]+matA[0][1]*matB[1][0];
	mat[0][1]=matA[0][0]*matB[0][1]+matA[0][1]*matB[1][1];
	mat[1][0]=matA[1][0]*matB[0][0]+matA[1][1]*matB[1][0];
	mat[1][1]=matA[1][0]*matB[0][1]+matA[1][1]*matB[1][1];

for(int i =0;i<2;i++)
{
	for(int j=0;j<2;j++)
		printf("%d ",mat[i][j]);
	printf("₩n");

}
	return 0;
}
```

27. 전역 변수와 지역 변수의 차이점에 대해 기술하시오.
전역변수 함수 바깥에 선언되며 어디에서든 접근가능 단 함수 밖에서 쓰기는 금지
지역변수 선언된 함수(해당 scope) 내에서만 사용이 가능하며 함수 호출 종료시 같이 사라짐
매개변수도 지역변수

28. 정수형 변수 2 개를 선언하고 서로 다른 값으로 초기화한다.
이후에 포인터를 사용하여 2 개의 값을 서로 교환해보라.

```
#include <stdio.h>

void swap(int *a, int *b)
{
	int temp;
	temp = *a;
	*a=*b;
	*b=temp;
}
```

```
int main(void)
{

	int A=1;
	int B=5;
	int *pA=&A;
	int *pB=&B;

	printf("%d,%d₩n",*pA,*pB);
	swap(pA,pB);
	printf("%d,%d₩n",*pA,*pB);
	return 0;
}
```

29. 임의의 구조체를 한 개 만들고 해당 구조체에 대한 변수 2 개를 선언한 이후
구조체의 멤버들에 적절한 값을 입력한 후 해당 값을 서로 교환해보라.

```
#include <stdio.h>

typedef struct{
	int x;
	int y;
}ST;



int main(void)
{
	ST A={1,2};
	ST B={3,4};
	ST C={0,0};

	printf("A: %d,%d₩n",A.x,A.y);
	printf("B: %d,%d₩n₩n",B.x,B.y);

	int temp=0;
	temp = A.x;
	A.x = B.x;
	B.x = temp;

	temp = A.y;
	A.y = B.y;
	B.y = temp;

	printf("A: %d,%d₩n",A.x,A.y);
```

```
        printf("B: %d,%d₩n",B.x,B.y);
        return 0;
}
```

30. 아래 코드를 보고 화면에 출력될 결과를 예측하시오.

```
#include <stdio.h>

int main(void)
{
        int shortcut1 = 0, shortcut2 = 3, num = 9;

        if(shortcut1 && num++) // 0 && 9  0 이므로 , if 문 건너뛴다 ——//shortcut1 이 이미 0 이므로 num++부분은 건너뛴다
                shortcut1++;                                          //따라서 num++는 실행되지 않는다.

        printf("%d, %d₩n", shortcut1, num); // 0, 9
                if(shortcut1 || shortcut2) // 0 || 3 1 이므로, if 문 안으로 들어간다
                        shortcut1++; // 1

        printf("%d₩n", shortcut1); // 1

        if(shortcut1 && ++num) // 1 && 11 1 이므로, if 문 들어간다
                shortcut2--; // 1

        printf("%d, %d, %d₩n", shortcut1, num, shortcut2); // 1  10  2
        return 0;
}
```

31. 아래 코드를 보고 화면에 출력될 결과를 예측하시오.

```
#include <stdio.h>

int main(void)
{
        int i, j;
        int num1, num2, num3;
        int arr[2][6]; // 2 행 6 열 배열선언
        int *arr_ptr[3] = { &num1, &num2, &num3 }; // int 형 포인터 3 개를 저장하는 배열
        int(*p)[6] = arr; //int 형 6 개 저장하는 배열을 가리키는 포인터
                          // *p 는 배열 arr 를 가리킨다.

        for (i = 0; i < 3; i++)
        {
                *arr_ptr[i] = i; //num1, num2, num3 에 각각 0,1,2 대입
                printf("%d₩n", *arr_ptr[i]); // 0,1,2 출력
        }
        for (i = 0; i < 2; i++)
                for (j = 0; j < 6; j++)
                        arr[i][j] = (i + 1) * (j + 1) * 10;
        /*      10 20 30 40  50  60
                20 40 60 80 100 120
                */

        for (i = 0; i < 2; i++)
                printf("%d₩n", *p[i]); // *p[0], *p[1] 출력: 10, 20

        for (i = 0; i < 2; i++)
                printf("%d₩n", *(p[i]+1)); // *(p[0]+1), *(p[1]+1) 출력: 20, 40
        return 0;
}
```

32. 아래 코드를 보고 해당 프로그램이 어떻게 동작하는지 기술하시오.

```
#include <stdio.h>

int fib(int num)
{
if(num == 1 || num == 2)
return 1;
else
return fib(num - 1) + fib(num - 2);
}

int main(void)
{
int result, final_val = 6;
result = fib(final_val);
printf("%d 번째 항의 수는 = %d₩n", final_val, result);
return 0;
}
```

피보나치 수열의 6 번째 항의 수를 구하는 프로그램이다.
fib 함수의 인자로 구하는 항의 수(n 항)를 넘겨주면
fib 함수 내에서 항의 수가 1 또는 2 일 경우 1 을 리턴하고 그 외의 숫자일 경우엔 n-1 항과 n-2 항에 대하여 재귀함수를 호출하고 해당하는 값을 반환한다.

33. 임의의 값 x 가 있는데, 이를 4096 단위로 정렬하고 싶다??면 어떻게 해야할까 ?
(힌트 : 4096 = 2^12)

34. 구조체를 사용하여 Stack 을 구현해보시오.

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

#define EMPTY 0

struct node{
        int data;
        struct node * link;
};

typedef struct node Stack;

Stack *get_node()
{
        Stack* tmp;
        tmp=(Stack*) malloc(sizeof(Stack));
        tmp->link = EMPTY;
        return tmp;
}

void Push(Stack** top, int data)
{
        Stack *tmp;
        tmp=*top; // 이전 노드 주소 저장
        *top=get_node(); // top 이 새 노드를 가리킨다.
        (*top)->data =data;
        (*top)->link = tmp;
}

int Pop(Stack** top)
{
        Stack* tmp;
        int num;
```

```
        tmp=*top; // 동적할당을 해제할 메모리 주소를 tmp에 저장해놓는다
        if(*top==EMPTY)
        {
                printf("Stack is EMPTY!₩n");
                return 0;
        }

        num=(*top)->data;
        *top =(*top)->link;
        free(tmp);
        return num;
}



int main(void)
{
        Stack*top=EMPTY;
        Push(&top,10); // top에서 Push하여 10을 넣는다
        Push(&top,20);
        Push(&top,30);
        printf("%d₩n",Pop(&top));
        printf("%d₩n",Pop(&top));
        printf("%d₩n",Pop(&top));
        printf("%d₩n",Pop(&top));

        return 0;

}
```

35. char *str = "Pointer Is Very Important"라는 문자열이 있다
    여기에 대문자가 총 몇 개 사용되었는지 세어보자

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

int main()
{
        char *str = "Pointer Is Very Important";
        int len = strlen(str);// 널문자 빼고 길이 구함
        int count = 0;
        for (int i = 0; i < len; i++)
        {
                if (str[i] <= 90 && str[i] !=' ')
                {
                        count++;
                        printf("%c₩n",str[i]);
                }
        }
        printf("총%d회₩n", count);
        return 0;
}
```

36. 포인터의 크기가 무엇에 따라 좌우되는지 기술하고 이유를 서술하시오.
메모리의 크기에 의해 좌우된다.
포인터는 메모리의 주소를 모두 나타낼 수 있어야 하므로
32bit 시스템에서 포인터의 크기는 32bit가 된다.

37. int p[4]와 int (*p)[4]의 차이에 대해 기술하시오.
Int 형 자료를 4개 저장하는 배열

Int 형 자료를 4 개 저장하는 배열을 가리키는 포인터

38. 함수 포인터를 사용하는 이유에 대해 기술하시오.
상황에 따라 다양한 함수를 호출 할 수 있으므로
특히 인터럽트 동작에 필수적

39. 아래의 선언을 보고 이것이 무엇인지 기술하시오.
(인자는 무엇이고 반환형은 무엇인지 함수인지 함수 포인터인지 등등)
void (* signal(int signum, void (*handler)(int)))(int);

void (*)(int); signal(int signum, void (*handler)(int))
반환형: void (*)(int)
- int 형을 인자로 받는 함수포인터
인자: int signum, void (*handler)(int)
-int 형 정수와 int 형을 인자로 받는 함수포인터

40. volatile 키워드의 용도는 무엇일까 ?
컴파일러에 의한 최적화 방지