TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
13일차 (2018. 03. 12)

4. goto문을 어떤 경우에 사용하면 효율적인지 작성하시오.

흔히, goto문은 초보자가 다루기 힘들어 사용하지 말라고 말하는 코드이다. 그러나 System Programming에서 배우겠지만 goto를 활용하면 특정한 상황을 아주 간결하게 해결할 수 있는 경우가 많다.

if 와 break 를 조합한 소스 코드와 goto 로 처리하는 소스코드를 가지고 있다고 가정해보자. for 문 수가 많을수록, break 거는 수도 많아진다(for에 break를 걸기 때문이다). break를 걸기 위해선 flag선언이 필요하다. 그리고 for문 끝마다 if(flag){ break; }를 해주어야 break를 할 수 있다. 또한, error가 났지만 코드는 계속 진행된다. 나중에 확인하고 error가 났기 때문에 사용할 수 없어 그냥 시간을 낭비하는 경우도 있다. 그러나 goto문을 사용하면 error가 난 순간에 멈출 수 있다. 즉, 가독성과 효율성 면에서 goto문이 더 좋다.

5. C언어에서 중요시 여기는 Memory 구조에 대해 기술하시오.

프로그램을 실행시키면 운영체제는 우리가 실행시킨 프로그램을 위해 메모리 공간을 할당한다. 메모리는 그 활용과 기능에 따라 가상(기능) 메모리로 Text, Data, Stack, Heap로 구분하기도 한다. 프로그래밍 실행 중에 수시로 메모리 할당과 반환이 이루어지는 경우 동적인 영역이라고 하며, 변함없이 프로그램이 끝날 때까지 계속 살아 있는 경우 정적인 영역이라고 한다.

Stack 은 함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역으로 함수 호출이 완료되면 사라진다.

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

Data 는 초기화 된 전역변수와 정적변수가 저장된 메모리 공간이고 프로그램의 시작과 동시에 할당되고, 프로그램이 종료되어야 메모리에서 소멸된다.

Text는 코드 세그먼트로 기계어로 변환된 코드가 저장 되어있다. 쓰기가 금지 되어있고 크기가 고정 되어있다.

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

스택 (메모리주소?)

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

```
#include <stdio.h>

int sum_2(int a, int b)
{ int sum = 0;
   int i;

      for(i=0; i < b; i++)
      if(i % 2 == 0)
      { sum += i;}
   return sum;
 }

int sum_1(int a, int b)
{ int sum = 0;
   int i;

      for(i = 0; i< b; i++)
      if(i % 3 == 0)
      { sum += i;}
   return sum;
}

int div(int a, int b)
{ double res = 0;
   int res_1 = 0;
            res_1 = sum_1(a, b) / sum_2(a, b);
            res = (int) res_1;
  return res;}

//a와 b는 범위를 설정한다.
int main (void)
{
            int a = 1;
            int b = 1000;

printf("%d ~ %d 사이의 홀수의 합은 %d이고 짝수의 합은 %d이다.\n",
a, b, sum_1(a,b), sum_2(a, b));
printf("그러므로, %d ~ %d사이의 홀수의 합을 짝수의 합으로 나눈 값은 %d이
다.\n", a, b, div(a, b));
return 0;
}
```

8. 7과 4이라는 2개의 정수를 입력받아 num1, num2에 저장하였다. num1 << num2, num1 >> num2의 값과 num1과 num2의 and, or, xor bit 연산을 수행하는 프로그램을 작성하시오.

```
#include <stdio.h>

int shift(int num1, int num2)
{ return (num1 << num2); }

int shift_1(int num1, int num2)
{ int res = 0;
    res = num1 >> num2;
return res;
}

int and(int num1, int num2)
{ return (num1 & num2); }

int or(int num1, int num2)
{ int res = 0;
   res = num1 | num2;
return res;
}

int xor(int num1, int num2)
{ return (num1 ^ num2); }

int main (void)
{
int a = 7;
int b = 4;

printf("%d << %d = %d\n", a, b, shift(a, b));
printf("%d >> %d = %d\n", a, b, shift_1(a, b));
printf("%d and %d = %d\n", a, b, and(a, b));
printf("%d or %d = %d\n", a, b, or(a, b));
printf("%d xor %d = %d\n", a, b, xor(a, b));

return 0;
}
```

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

디버깅은 컴파일은 성공적으로 되었으나(즉 문법 오류 없음), 논리적인 오류가 존재하는 경우에 수행하는 것이다.

그 외에도 예측치 못한 다양한 문제가 존재할 수 있다. 이럴 경우에 무엇이 문제인지 파악하기 위해 디버깅을 하는 것이다. 즉, 프로그램이 동작은 하는데 정상적으로 완료하지 못하고, 왜 이상한 동작을 하는지 알고자 할 때 하는 것이 디버깅이다.

[디버깅 하는 순서] gcc -g ~~~.c > gdb a.out > ~ (gdb) 이 뜬다. > b main > r

10-11번 복합문제

10. do while문을 사용하는 이유에 대해 기술하시오.

일단 조건이 만족하지 않더라도 1번은 수행한다. 즉, 무조건 1번은 실행하게 되어 있는 함수이다. 궁극적으로 do while을 사용하는 이유는 매크로 확장 때문이다. Kernel의 매크로에 자주 사용된다.

11. 표준 상태에서 아래의 소스 코드가 컴파일이 되는지 안되는지 판정하시오.

만약 컴파일이 안된다면 어떻게 바꿔야 하는지 기술하시오.

```
#define inc_each(x, y) { x++; y++; }

int main(void)
{
int x = 10, y = 5;

if(x > y)
inc_each(x, y);
else
x = y;

return 0;
}
```

12. 다음 빈칸을 채우시오.
컴퓨터 구조상 가장 빠른 것은 ()이고 가장 느린것은()이다.
> 레지스터가 빠르고 디스크가 가장 느리다.

13. 이중 for문을 사용하여 구구단을 작성해보시오.

```
int mul(void)
{
	int a;
	int b;

	for(a=1; a<10; a++)
	{
		for(b=1; b<10; b++)
		{
		   printf("%d * %d = %d\t", a , b, a*b);
		}
	}
	return 0;
}

int main(void)
{
mul();
return 0;
}
```

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;
```

}

if문이 만족할 때만 빠져나올 수가 있는데 여기서 number는 1000이 될 수가 없다. 그래서 if (number <= 1000) 으로 바꾸어주어야 한다.

15. 현재 디버깅중이며 메모리 주소 0x7ffffb70에 어떤 값이 저장되었다. 이 때 위 메모리에 저장된 값을 보고 싶다면 어떻게 해야 하는가 ?

16. 다음 어셈블리어를 보고 함수의 main함수부터 my_function 함수까지 stack의 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>

int main(void)
{
   int i;
   char str[50];
   printf("문자열을 입력하세요: ");
   scanf("%s",str);
   printf("홀수 문자는 ");
      for(i = 0; i < 50; i++)
      {
      if(str[i]=='\0')
      break;
      if(i % 2 == 0)
      printf("%c", str[i]);
      }
      printf("\n");
      return 0;
}
```

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

19. 구조체를 사용하는 이유에 대하여 기술하라.
  > 자료를 처리하다 보니 하나로 묶어야 편하기 때문이다. 문자열과, 숫자를 한 번에 묶어서 관리하고 싶을 때 등 어떠한 정보와 그 연결 된 정보를 관리하고 싶을 때 쓴다. 즉, 관련 정보를 하나의 의미로 묶을 때 쓴다.

20-22번 복합 문제

20. 20개의 배열에 rand()를 이용하여 무작위로 1 ~ 100까지의 값을 저장하는 프로그램을 작성하

라.
21. 그리고 각 배열의 요소의 홀수들과 짝수들의 합을 구해 출력하라.
22. 홀수들의 합과 짝수들의 합을 곱하여 출력하라.

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

24-25 복합 문제
24. 어떤 정수에 값 13이 들어 있다. 이것을 4중 포인터를 사용하여 표현해보라.
25. 4중 포인터를 사용하여 표현했다면 현재 값인 13을 14로 증가시켜보자. (num++을 사용하지 말라)

26. 행렬의 곱셈을 프로그램으로 작성하라.

27. 전역 변수와 지역 변수의 차이점에 대해 기술하시오.
전역변수 함수 바깥에 선언되며 어디에서든 접근 가능하다. 메모리 구조에서 data에 존재한다.
지역변수 선언된 함수(해당 scope) 내에서만 사용이 가능하며 함수 호출 종료시 같이 사라진다. 지역변수는 stack에 위치한다. 매개변수도 지역변수다.

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

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

30. 아래 코드를 보고 화면에 출력될 결과를 예측하시오.

```
#include< stdio.h>

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

if(shortcut1 && num++)
shortcut1++;

printf("%d, %d\n", shortcut1, num)

if(shortcut1 || shortcut2)
shortcut1++;

printf("%d\n", shortcut1);

if(shortcut1 && ++num)
shortcut2--;

printf("%d, %d, %d\n", shortcut1, num, shortcut2);

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];
int *arr_ptr[3] = {&num1, &num2, &num3};
int (*p)[6] = arr;

for(i = 0; i < 3; i++)
{
*arr_ptr[i] = i;
printf("%d\n", *arr_ptr[i]);
}
for(i = 0; i < 2; i++)
for(j = 0; j < 6; j++)
arr[i][j] = (i + 1) * (j + 1) * 10;

for(i = 0; i < 2; i++)
printf("%d\n", *p[i]);

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("%d번째 항의 수는 = %d\n", final_val, result);
return 0;
}
```

33. 임의의 값 x가 있는데, 이를 4096 단위로 정렬하고 싶다면 어떻게 해야 할까? (힌트 : 4096 = 2^12)

34. 구조체를 사용하여 Stack을 구현해보시오.

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

36. 포인터의 크기가 무엇에 따라 좌우되는지 기술하고 이유를 서술하시오.
> 포인터는 컴퓨터의 HW에 따라 달라진다. 64Bit 시스템이면 포인터는 8byte이고 32bit면 4byte가 된다. 최대값이 되는 이유는 만약 64비트 시스템에서 8byte가 아닌 4byte가 될 경우, 그 보다 큰 데이터형이 오면 온전하게 저장할 수 없어 값이 달라지기 때문이다.

37. int p[4]와 int (*p)[4]의 차이에 대해 기술하시오.

Int p[4]는 int 형 데이터를 4개 저장 가능한 배열 p를 뜻하는 것이고int(*p)[4] 는 int형 데이터 4개를 저장하는 배열을 가르키는 포인터다.

38. 함수 포인터를 사용하는 이유에 대해 기술하시오.
>함수를 인자로 사용할 때 함수 포인터를 이용하여 사용한다.

39. 아래의 선언을 보고 이것이 무엇인지 기술하시오. (인자는 무엇이고 반환형은 무엇인지 함수인지 함수 포인터인지 등등)

void (* signal(int signum, void (*handler)(int)))(int);

40. volatile 키워드의 용도는 무엇일까 ?