TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 윤연성
whatmatters@naver.com

```
/*

#include <stdio.h>

int main(void)
{
   int arr[3] = {1, 2, 3};
   int *p = arr; // 포인터로 쓸때 *p = arr 로 가능!
   int i;

   for(i = 0; i < 3; i++)
      printf("p[%d] = %d\n", i, p[i]); //

   return 0;
}


#include <stdio.h>
int main(void)
{
int arr[4] = {10, 20, 30, 40}; //배열 arr 에 값 넣기
int i;
printf("address = %p\n", arr); //배열만들면 자체가 주소값
for (i = 0; i < 4; i++)
{
printf("address = %p, arr[%d] = %d\n", &arr[i], i, arr[i]); // arr 이나 &arr[0]과 같음 **
}

return 0;
}


#include <stdio.h>

void add_arr(int *arr) //배열자체가 포인터임**
{
   int i;
   for(i = 0; i < 3; i++)
   {
      arr[i] += 7; // arr[i] = arr[i] + 7
   }
}
void print_arr(int *arr) //배열자체가 포인터 **
{
   int i;
   for(i = 0; i < 3; i++)
   {
      printf("arr[%d] = %d\n", i, arr[i]);
   }
}
int main(void)
{
```

```
    int i;
    int arr[3] = {1, 2, 3};
    add_arr(arr);
    print_arr(arr);

    printf("real\n");

    for(i = 0; i < 3; i++)
    {
        printf("arr[%d] = %d\n", i, arr[i]);
    }

    return 0;
}

// 수정

int kinds_of_rot(avl *root, int data)
{
//for RR and RL
//if(rotation_check(root) >1)
if(rotation_check(root) >1)
{
if(rotation_check(root->right) <0)
return RL;
return RR;
}

else if(rotation_check(root) <-1)
{
if(rotation_check(root->left)>0)
return LR;
return LL;
}
}



void avl_ins(avl **root, int data)
{
int cnt = 0;
avl **tmp = root;
stack *top = NULL;
//push(&top, *tmp);
while(*tmp)
{
printf("Save Stack: %d, data = %d\n", ++cnt, data);
//push(&top, *tmp);
push(&top, tmp)

if((*tmp) ->data > data)
tmp = &(*tmp) ->letf;
```

```
else if((*tmp) ->data < data)
tmp = &(*tmp) -> right;
}
*tmp = get_avl_node();
(*tmp) ->data = data;

while(stack_is_not_empty(top))
{
printf("Extract Stack : %d, data = %d\n", --cnt, data);
//need to change here with pointer of pointer
avl **t = (avl **)pop(&top);
(*t) -> lev = update_level(*t);
if(abs(rotation_check(*t)) > 1)
{
printf("Insert Rotation\n");
*t = rotation(*t, kincds_of_rot(*t,data));
}
}
}
```

```
#include <stdio.h>

int main(void)
{
   int num = 3;
   int *p = &num; //num 의 주소값
   int **pp = &p; // **포인터 는 이중포인터로 &p 주소값이 들어가야됨
   int ***ppp = &pp; // ***는 삼중포인터로 **pp 의 주소값이 들어감

   printf("num = %d\n", num);
   printf("*p = %d\n", *p); //*p 가 그리키는 주소로 간후 데이터값 출력
   printf("**pp = %d\n", **pp); // **pp 가 가르키는 주소로 간후 데이터값 출력
   printf("***ppp = %d\n", ***ppp);

   return 0;
}
```

```
// int a[] = {1, 2, 3, 4, 5};
int *p;

p = a; // 요기서 a 는 배열의 시작하는 주소값을 말해주고 p 에 넣음
```

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

```
typedef struct __queue
{
   int data;
   struct __queue *link;
} queue;

queue *get_node(void)
{
   queue *tmp;
   tmp = (queue *)malloc(sizeof(queue));
   tmp->link = NULL;
   return tmp;
}

void enqueue(queue **head, int data)
{
   if(*head == NULL)
   {
      *head = get_node();
      (*head)->data = data;
      return;
   }

   enqueue(&(*head)->link, data);
}

void print_queue(queue *head)
{
   queue *tmp = head;

   while(head)
   {
      printf("head->data = %d\n", head->data);
      head = head->link;
   }
}

void dequeue(queue **head, int data)
{
   queue *tmp = *head;

   if(*head == NULL)
      printf("There are no data that you delete\n");

   if((*head)->data != data)
      dequeue(&(*head)->link, data);
   else
   {
      //queue *res = head->link;
      printf("Now you delete %d\n", data);
      *head = tmp->link;
```

```
      free(tmp);
   }
}

int main(void)
{
   int i;

   queue *head = NULL;

   srand(time(NULL));

   for(i = 0; i < 3; i++)
      enqueue(&head, (i + 1) * 10);

   print_queue(head);

   dequeue(&head, 20);

   print_queue(head);

   return 0;
}


#include <stdio.h>

int main(void)
{

int arr[7] = {10, 20, 30};
int *arr_p = &arr[1];

printf("현재값=%d\n", *arr_p);
printf("한칸 + 이동=%d\n", *(arr_p + 1));
printf("한칸 -이동=%d\n", *(arr_p - 1));
printf("세칸 + 이동=%d\n", *(arr_p + 3));

printf("%d\n",arr_p);
printf("%d\n",arr);

return 0;

}


#include <stdio.h>

int main(void)
{
int arr[3][3];
// int *p = arr;
```

```
// *p = arr;
int i, j;

for(i = 0; i<3; i++ )
{
for(j =0; j<3; j++)
{
printf("%d\n", &arr[i][j]); //배열에서 arr 자체가 주소값인데 arr[i][j]하면 이상한값이 나옴
}
}

return 0;
}
```

```
//1 번

#include <stdio.h>

int main(void)

{

char chr;

scanf("%c",&chr);
printf("chr = %c\n", chr^32); //아스키코드 값참조 10 진수값으로 32 차이남

return 0;
}
```

```
//3 번

#include <stdio.h>

int main(void)

{
int i;
int x;
scanf("%d", &x); //임의의값 x 를 입력받음

for(i= 0; i < x; i++)
{
int a[x]; //임의의값의 배열을 만듬

printf("a[%d]=%d\n",i, &a[i]); //int 배열과 포인터는 4 바이트씩 할당
```

```
}

//printf("%d\n", sizeof(a)); //sizeof()자료형의 크기를 알수있음

return 0;

}

// 4 번

int p[7] 와 int (*p)[7] 가 있다.

int p[7]= p[0], p[1], p[2], p[3], p[4], p[5], p[6] 그냥 배열임 주소값은 0 4 8
int (*p)[7] = 포인터 배열이며 포인터를 담고있는 배열이다.(주소값이 들어간다)

// 7 번

헤더파일

// 9 번

레지스터
캐시
메모리
디스크

// 10 번
스택 = 지역변수가 위치하는 영역
힙 = 동적 할당된 녀석들이 위치하는 영역
데이터 = 전역변수 및 static 으로 선언된 것들이 위치하는 영역
캐시 = Machine Code 가 위치하는 영역


//12 번

void (* signal(int signum, void (* handler)(int)))(int)
프로토타입 반환형, 함수의이름 , 인자(파라미터)




*/
```