



รายงาน

เรื่อง การทำงานของอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายงาน

เรื่อง การทำงานของอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ENGCE124 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures and Algorithms) หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่ ในระดับปริญญาตรีปีที่ 2 โดยมีจุดประสงค์ในอธิบายการทำงานของอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension] ในส่วนของโค้ดการทำงานของโปรแกรม ไม่ว่าจะเป็นโค้ดในส่วนอาร์เรย์ 1 มิติ อาร์เรย์ 2 มิติ และ อาร์เรย์ 3 มิติ โค้ดคำสั่งโดยรวมทั้งหมดของโปรแกรม รวมถึงผลลัพธ์การแสดงผลของโปรแกรม

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาการทำงานของอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension] หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใดผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 18/07/2567

สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรมอาเรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]	1
การอธิบายโค้ดโปรแกรมอาเรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension] แบบแยกย่อย	4
หลักการทำงานของโปรแกรมอาเรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]	8
ผลลัพธ์การทำงานของโปรแกรมอาเรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]	9
บรรณานุกรม	12

โค้ดของโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]

```

/* Program create array 1-3 dimension in function by Pongpan Laowaphong 66543206019-2
1. Calculate and Allocate memory
2. Calculate the memoryaddress of array
3. Use point directed in to memory and read/write its
4. Formular useing
Element = (u-l+1)
Element = (u1-l1+1)*(u2-l2+1)
Element = (u1-l1+1)*(u2-l2+1)*(u3-l3+1)
Total_mem = Element*C
Address of Array
A(i) =BA+(i-l)C
A(i,j) =BA+(i-l1)*(u2-l2+1)C+(j-l2)C
A(i,j,k)=BA+(i-l1)*(u2-l2+1)(u3-l3+1)C+(j-l2)(u3-l3+1)C+(k-l3)C
=====*/

#include <stdio.h> //use printf()
#include <conio.h> //use getch()
#include <stdlib.h> //use malloc()

#define l 1 //lower Bound
#define u 5 //Upper Bound
#define l1 1 //lower Bound 1
#define u1 3 //Upper Bound 1
#define l2 1 //Lower Bound 2
#define u2 4 //Upper Bound 2
#define l3 1 //Lower Bound 3
#define u3 5 //Upper Bound 3

int *BA1, *BA2, *BA3, *p; //Base address of each dimension and moving pointer
int i,j,k; //subscript of Array

void Create1DArray(){ //Create Array 1 dimension
    int element,c,total_mem; //Variable uses
    element=(u-l+1); //Calculate element
    c=sizeof(*BA1); //Calculate Size each block of Array
    total_mem=element*c; //Calculate Total Size

```

```

        BA1=(int*)malloc(total_mem); //Memory allocate and use BA1 point its
    }

    void A1(int i,int x){ //Put data into Array 1 Dimension
        p=BA1+(i-l); //Calculate pointer
        *p=x; //Put data
    }

    int ReadA1(int i) { //Read data from Array 1 Dimension
        p=BA1+(i-l); //Calculate pointer
        return(*p); //Return value in Array 28
    }

    //-----

    void Create2DArray() {
        int element,c,total_mem;
        element=(u1-l1+1)*(u2-l2+1);
        c=sizeof(*BA2);
        total_mem=element*c;
        BA2=(int*)malloc(total_mem);
    }

    void A2(int i,int j,int x) {
        p=BA2+((i-l1)*(u2-l2+1)+(j-l2));
        *p=x;
    }

    int ReadA2(int i,int j) {
        p=BA2+(i-l1)*(u2-l2+1)+(j-l2);
        return(*p);
    }

    //-----

    void Create3DArray() {
        int element,c,total_mem;
        element=(u1-l1+1)*(u2-l2+1)*(u3-l3+1);
        c=sizeof(*BA3);
        total_mem=element*c;
        BA3=(int*)malloc(total_mem);
    }

```

```

void A3(int i,int j,int k,int x) {
    p=BA3+((i-l1)*(u2-l2+1)*(u3-l3+1)+(j-l2)*(u3-l3+1)+(k-l3));
    *p=x;
}

int ReadA3(int i,int j,int k){
    p=BA3+(i-l1)*(u2-l2+1)*(u3-l3+1)+(j-l2)*(u3-l3+1)+(k-l3); ;
    return(*p);
}

//-----

int main() {
    printf("1-3 DIMENSION ARRAY FUNCTION...\n");
    printf("=====\n");
    // Create Array.....
    Create1DArray();
    Create2DArray();
    Create3DArray();
    //Using 1 Dimention Array...
    i=2;
    A1(i,9);
    printf("\nA1(%d) = %d ",i,ReadA1(i));
    //Using 2 Dimension Array ... 29
    i=2; j=3;
    A2(i,j,99);
    printf("\nA2(%d,%d) = %d ",i,j,ReadA2(i,j));
    //Using 3 Dimension Array...
    i=3; j=4;k=5;
    A3(i,j,k,999);
    printf("\nA3(%d,%d,%d) = %d ",i,j,k,ReadA3(i,j,k));
    getch(); //Wait for KBD hit
    free(BA1); //Free memory of each array
    free(BA2);
    free(BA3);
    return(0);
} //End MAIN Fn.

```

การอธิบายโค้ดโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension] แบบแยกย่อย

โค้ดโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ มีวัตถุประสงค์เพื่อสร้างและจัดการอาร์เรย์ในรูปแบบ 1-3 มิติ โดยมีการคำนวณและจัดสรรหน่วยความจำให้กับอาร์เรย์ และใช้ตัวชี้ (pointer) เพื่ออ่านและเขียนค่าลงในหน่วยความจำที่จัดสรรนั้น โดยโค้ดแต่ละส่วนประกอบไปด้วย

1. ส่วนประกาศตัวแปรและไลบรารี

```
#include <stdio.h> // ไลบรารีสำหรับใช้ฟังก์ชัน printf()
#include <conio.h> // ไลบรารีสำหรับใช้ฟังก์ชัน getch()
#include <stdlib.h> // ไลบรารีสำหรับใช้ฟังก์ชัน malloc()

#define l 1 // กำหนดขอบเขตล่าง
#define u 5 // กำหนดขอบเขตบน

#define l1 1 // กำหนดขอบเขตล่างของมิติที่ 1
#define u1 3 // กำหนดขอบเขตบนของมิติที่ 1
#define l2 1 // กำหนดขอบเขตล่างของมิติที่ 2
#define u2 4 // กำหนดขอบเขตบนของมิติที่ 2
#define l3 1 // กำหนดขอบเขตล่างของมิติที่ 3
#define u3 5 // กำหนดขอบเขตบนของมิติที่ 3

int *BA1, *BA2, *BA3, *p; // ตัวชี้ไปยังที่อยู่หน่วยความจำฐานของแต่ละมิติและตัวชี้ที่เคลื่อนที่
int i, j, k; // ตัวแปรดัชนีของอาร์เรย์
```

2. ฟังก์ชันสร้างอาร์เรย์ 1 มิติ

```
void Create1DArray() {
    int element, c, total_mem; // ตัวแปรใช้ในการคำนวณ
    element = (u - l + 1); // คำนวณจำนวนองค์ประกอบ
    c = sizeof(*BA1); // คำนวณขนาดของแต่ละบล็อกในอาร์เรย์
    total_mem = element * c; // คำนวณขนาดรวมของหน่วยความจำที่ต้องการ
    BA1 = (int*)malloc(total_mem); // จัดสรรหน่วยความจำและใช้ BA1 ชี้ไปยังหน่วยความจำนั้น
}
```


3. ฟังก์ชันใส่ค่าและอ่านค่าในอาร์เรย์ 1 มิติ

```
void A1(int i, int x) {
    p = BA1 + (i - l); // คำนวณตำแหน่งของตัวชี้
    *p = x; // ใส่ค่า
}

int ReadA1(int i) {
    p = BA1 + (i - l); // คำนวณตำแหน่งของตัวชี้
    return (*p); // คืนค่าจากอาร์เรย์
}
```

4. ฟังก์ชันสร้างอาร์เรย์ 2 มิติ

```
void Create2DArray() {
    int element, c, total_mem;

    element = (u1 - l1 + 1) * (u2 - l2 + 1); // คำนวณจำนวนองค์ประกอบ
    c = sizeof(*BA2); // คำนวณขนาดของแต่ละบล็อกในอาร์เรย์
    total_mem = element * c; // คำนวณขนาดรวมของหน่วยความจำที่ต้องการ
    BA2 = (int*)malloc(total_mem); // จัดสรรหน่วยความจำและใช้ BA2 ชี้ไปยังหน่วยความจำนั้น
}
```

5. ฟังก์ชันใส่ค่าและอ่านค่าในอาร์เรย์ 2 มิติ

```
void A2(int i, int j, int x) {
    p = BA2 + ((i - l1) * (u2 - l2 + 1) + (j - l2)); // คำนวณตำแหน่งของตัวชี้
    *p = x; // ใส่ค่า
}

int ReadA2(int i, int j) {
    p = BA2 + (i - l1) * (u2 - l2 + 1) + (j - l2); // คำนวณตำแหน่งของตัวชี้
    return (*p); // คืนค่าจากอาร์เรย์
}
```

6. ฟังก์ชันสร้างอาร์เรย์ 3 มิติ

```
void Create3DArray() {
    int element, c, total_mem;

    element = (u1 - l1 + 1) * (u2 - l2 + 1) * (u3 - l3 + 1); // คำนวณจำนวนองค์ประกอบ
    c = sizeof(*BA3); // คำนวณขนาดของแต่ละบล็อกในอาร์เรย์
    total_mem = element * c; // คำนวณขนาดรวมของหน่วยความจำที่ต้องการ
    BA3 = (int*)malloc(total_mem); // จัดสรรหน่วยความจำและใช้ BA3 ชี้ไปยังหน่วยความจำนั้น
}
```

7. ฟังก์ชันใส่ค่าและอ่านค่าในอาร์เรย์ 3 มิติ

```
void A3(int i, int j, int k, int x) {
    p = BA3 + ((i - l1) * (u2 - l2 + 1) * (u3 - l3 + 1) + (j - l2) * (u3 - l3 + 1) + (k - l3)); // คำนวณตำแหน่งของตัวชี้
    *p = x; // ใส่ค่า
}

int ReadA3(int i, int j, int k) {
    p = BA3 + (i - l1) * (u2 - l2 + 1) * (u3 - l3 + 1) + (j - l2) * (u3 - l3 + 1) + (k - l3); // คำนวณตำแหน่งของตัวชี้
    return (*p); // คืนค่าจากอาร์เรย์
}
```

8. ฟังก์ชันหลัก

```
int main() {
    printf("1-3 DIMENSION ARRAY FUNCTION...\n");
    printf("=====\n");
    // สร้างอาร์เรย์
    Create1DArray();
    Create2DArray();
    Create3DArray();
    // ใช้อาร์เรย์ 1 มิติ
    i = 2;
    A1(i, 9);
    printf("\nA1(%d) = %d ", i, ReadA1(i));
}
```

8. ฟังก์ชันหลัก (ต่อ)

```
// ใช้อาร์เรย์ 2 มิติ
i = 2; j = 3;
A2(i, j, 99);
printf("\nA2(%d, %d) = %d ", i, j, ReadA2(i, j));
// ใช้อาร์เรย์ 3 มิติ
i = 3; j = 4; k = 5;
A3(i, j, k, 999);
printf("\nA3(%d, %d, %d) = %d ", i, j, k, ReadA3(i, j, k));
getch(); // รอการกดปุ่มใดๆ
// คืนหน่วยความจำของแต่ละอาร์เรย์
free(BA1);
free(BA2);
free(BA3);
return (0);
}
```

หลักการทำงานของโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]

โปรแกรมโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension] มีวัตถุประสงค์เพื่อสร้างและจัดการอาร์เรย์แบบ 1 มิติ, 2 มิติ และ 3 มิติ โดยใช้การคำนวณและจัดสรรหน่วยความจำ พร้อมทั้งใช้ตัวชี้ (pointer) ในการเข้าถึงและแก้ไขค่าในอาร์เรย์เหล่านั้น โดยในการสร้างอาร์เรย์ โปรแกรมเริ่มต้นด้วยการประกาศขอบเขตของอาร์เรย์ในแต่ละมิติ เช่น ขอบเขตล่างและบนของอาร์เรย์ 1 มิติ, 2 มิติ และ 3 มิติ การประกาศขอบเขตเหล่านี้ช่วยให้โปรแกรมสามารถคำนวณจำนวนองค์ประกอบของอาร์เรย์และจัดสรรหน่วยความจำได้อย่างถูกต้อง ซึ่งฟังก์ชันสำหรับการสร้างอาร์เรย์จะคำนวณขนาดของอาร์เรย์และจำนวนหน่วยความจำที่ต้องใช้ จากนั้นจะจัดสรรหน่วยความจำให้กับอาร์เรย์แต่ละมิติ ตัวอย่างเช่น

- ฟังก์ชัน Create1DArray() สร้างอาร์เรย์ 1 มิติ
- ฟังก์ชัน Create2DArray() สร้างอาร์เรย์ 2 มิติ
- ฟังก์ชัน Create3DArray() สร้างอาร์เรย์ 3 มิติ

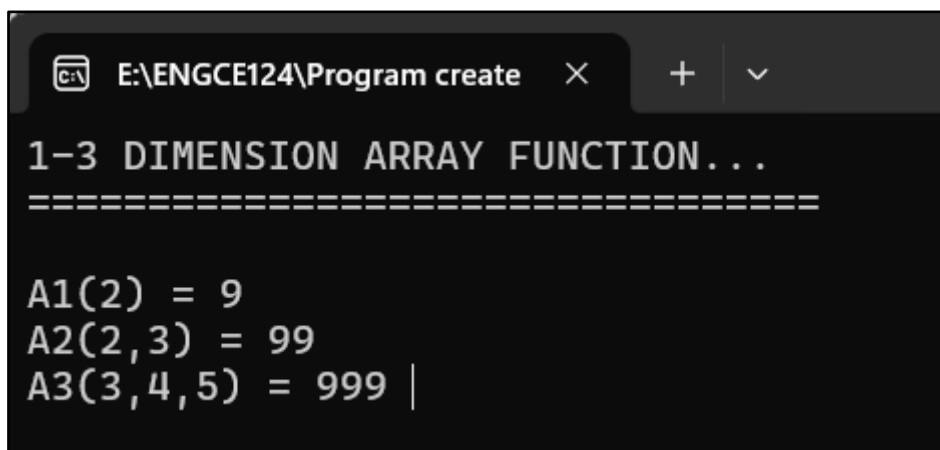
ในส่วนของการใส่ค่าและอ่านค่า โปรแกรมมีฟังก์ชันสำหรับใส่ค่า (Write) และอ่านค่า (Read) ในอาร์เรย์แต่ละมิติ ฟังก์ชันเหล่านี้ใช้ตัวชี้ (pointer) เพื่อคำนวณตำแหน่งหน่วยความจำที่ต้องการใส่หรืออ่านค่า ตัวอย่างเช่น

- ฟังก์ชัน A1(int i, int x) ใช้ในการใส่ค่า x ลงในตำแหน่งที่ i ของอาร์เรย์ 1 มิติ
- ฟังก์ชัน ReadA1(int i) ใช้ในการอ่านค่าจากตำแหน่งที่ i ของอาร์เรย์ 1 มิติ
- ฟังก์ชัน A2(int i, int j, int x) และ ReadA2(int i, int j) ใช้ในการใส่ค่าและอ่านค่าจากอาร์เรย์ 2 มิติ
- ฟังก์ชัน A3(int i, int j, int k, int x) และ ReadA3(int i, int j, int k) ใช้ในการใส่ค่าและอ่านค่าจากอาร์เรย์ 3 มิติ

ในส่วนของฟังก์ชันหลักของโปรแกรมมีหน้าที่ในการเรียกใช้ฟังก์ชันต่างๆ เพื่อสร้างอาร์เรย์ ใส่ค่าและอ่านค่าจากอาร์เรย์ และแสดงผลลัพธ์ นอกจากนี้ยังมีการคืนหน่วยความจำที่จัดสรรให้เมื่อการทำงานเสร็จสมบูรณ์ โดยหลังจากทำงานเสร็จ โปรแกรมจะคืนหน่วยความจำที่จัดสรรให้โดยใช้ฟังก์ชัน free()

สรุปแล้วโปรแกรมโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension] แสดงให้เห็นถึงการสร้างและจัดการอาร์เรย์หลายมิติในภาษา C โดยใช้การคำนวณและการจัดสรรหน่วยความจำอย่างมีประสิทธิภาพ นอกจากนี้ยังแสดงให้เห็นถึงการใช้งานตัวชี้ในการเข้าถึงและแก้ไขค่าในอาร์เรย์ ซึ่งเป็นพื้นฐานสำคัญในการเขียนโปรแกรมเชิงโครงสร้างและการจัดการหน่วยความจำในภาษา C

ผลลัพธ์การทำงานของโปรแกรมอาร์เรย์ 1 ถึง 3 มิติ [Array 1-3 Dimension]



```

E:\ENGCE124\Program create  X  +  v
1-3 DIMENSION ARRAY FUNCTION...
=====

A1(2) = 9
A2(2,3) = 99
A3(3,4,5) = 999 |
  
```

ในผลลัพธ์ข้างต้นสามารถอธิบายการทำงานโดยอ้างอิงจากผลลัพธ์ได้ดังนี้

1. การเริ่มต้นการทำงานของโปรแกรม

- โปรแกรมแสดงข้อความ "1-3 DIMENSION ARRAY FUNCTION..." และ "=====" เพื่อบ่งบอกถึงการทำงานเกี่ยวกับอาร์เรย์หลายมิติ

2. การสร้างอาร์เรย์ 1 มิติ

- เรียกใช้ฟังก์ชัน Create1DArray()
 - คำนวณจำนวนองค์ประกอบของอาร์เรย์ : $\text{element} = (u-l+1) = (5-1+1) = 5$
 - คำนวณขนาดของหน่วยความจำที่ต้องจัดสรร :
 $\text{total_mem} = \text{element} \times \text{sizeof}(*BA1) = 5 \times 4 = 20 \text{ ไบต์ (สำหรับตัวแปรประเภท int)}$
 - จัดสรรหน่วยความจำให้กับอาร์เรย์ 1 มิติ

3. การใส่ค่าและอ่านค่าในอาร์เรย์ 1 มิติ

- ใส่ค่า 9 ที่ตำแหน่ง 2 โดยใช้ฟังก์ชัน A1(i, x):
 - คำนวณตำแหน่งในหน่วยความจำ : $p = BA1 + (i-l) = BA1 + (2-1) = BA1+1$
 - ใส่ค่า 9 ที่ตำแหน่งดังกล่าว
- อ่านค่าที่ตำแหน่ง 2 โดยใช้ฟังก์ชัน ReadA1(i):
 - คำนวณตำแหน่งในหน่วยความจำ : $p = BA1 + (i-l) = BA1 + (2-1) = BA1+1$
 - อ่านค่าและคืนค่า 9
- แสดงผล: "A1(2) = 9"

4. การสร้างอาร์เรย์ 2 มิติ

- เรียกใช้ฟังก์ชัน `Create2DArray()`
 - คำนวณจำนวนองค์ประกอบของอาร์เรย์ :
$$\text{element} = (u1-l1+1) \times (u2-l2+1) = (3-1+1) \times (4-1+1) = 3 \times 4 = 12$$
 - คำนวณขนาดของหน่วยความจำที่ต้องจัดสรร :
$$\text{total_mem} = \text{element} \times \text{sizeof}(*\text{BA2}) = 12 \times 4 = 48 \text{ ไบต์}$$
 - จัดสรรหน่วยความจำให้กับอาร์เรย์ 2 มิติ

5. การใส่ค่าและอ่านค่าในอาร์เรย์ 2 มิติ

- ใส่ค่า 99 ที่ตำแหน่ง (2, 3) โดยใช้ฟังก์ชัน `A2(i, j, x)` :
 - คำนวณตำแหน่งในหน่วยความจำ : $p = \text{BA2} + ((i-l1) \times (u2-l2+1) + (j-l2)) = \text{BA2} + ((2-1) \times 4 + (3-1)) = \text{BA2} + (1 \times 4 + 2) = \text{BA2} + 6$
 - ใส่ค่า 99 ที่ตำแหน่งดังกล่าว
- อ่านค่าที่ตำแหน่ง (2, 3) โดยใช้ฟังก์ชัน `ReadA2(i, j)` :
 - คำนวณตำแหน่งในหน่วยความจำ : $p = \text{BA2} + ((i-l1) \times (u2-l2+1) + (j-l2)) = \text{BA2} + ((2-1) \times 4 + (3-1)) = \text{BA2} + (1 \times 4 + 2) = \text{BA2} + 6$
 - อ่านค่าและคืนค่า 99
- แสดงผล: **"A2(2,3) = 99"**

6. การสร้างอาร์เรย์ 3 มิติ

- เรียกใช้ฟังก์ชัน `Create3DArray()`
 - คำนวณจำนวนองค์ประกอบของอาร์เรย์ : $\text{element} = (u1-l1+1) \times (u2-l2+1) \times (u3-l3+1) = (3-1+1) \times (4-1+1) \times (5-1+1) = 3 \times 4 \times 5 = 60$
 - คำนวณขนาดของหน่วยความจำที่ต้องจัดสรร :
$$\text{total_mem} = \text{element} \times \text{sizeof}(*\text{BA3}) = 60 \times 4 = 240 \text{ ไบต์}$$
 - จัดสรรหน่วยความจำให้กับอาร์เรย์ 3 มิติ

7. การใส่ค่าและอ่านค่าในอาร์เรย์ 3 มิติ

- ใส่ค่า 999 ที่ตำแหน่ง (3, 4, 5) โดยใช้ฟังก์ชัน `A3(i, j, k, x)`:
 - คำนวณตำแหน่งในหน่วยความจำ : $p = \text{BA3} + ((i-l1) \times (u2-l2+1) \times (u3-l3+1) + (j-l2) \times (u3-l3+1) + (k-l3)) =$

$$BA3 + ((3-1) \times 4 \times 5 + (4-1) \times 5 + (5-1)) = BA3 + (2 \times 20 + 3 \times 5 + 4) =$$

$$BA3 + (40 + 15 + 4) = BA3 + 59$$

○ ใส่ค่า 999 ที่ตำแหน่งดังกล่าว

- อ่านค่าที่ตำแหน่ง (3, 4, 5) โดยใช้ฟังก์ชัน ReadA3(i, j, k) :

○ คำนวณตำแหน่งในหน่วยความจำ :

$$p = BA3 + ((i-l1) \times (u2-l2+1) \times (u3-l3+1) + (j-l2) \times (u3-l3+1) + (k-l3)) =$$

$$BA3 + ((3-1) \times 4 \times 5 + (4-1) \times 5 + (5-1)) = BA3 + (2 \times 20 + 3 \times 5 + 4) =$$

$$BA3 + (40 + 15 + 4) = BA3 + 59$$

○ อ่านค่าและคืนค่า 999

- แสดงผล: "A3(3,4,5) = 999"

8. การคืนหน่วยความจำ

- หลังจากทำงานเสร็จ โปรแกรมจะคืนหน่วยความจำที่จัดสรรให้โดยใช้ฟังก์ชัน free() สำหรับแต่ละอาร์เรย์ ได้แก่ BA1, BA2, และ BA3

บรรณานุกรม

ChatGPT. (-). การสร้างและจัดการอาร์เรย์หลายมิติในภาษา C. สืบค้น 18 กรกฎาคม 2567,
จาก <https://chatgpt.com/>