**Note : Câu 7 và 6 làm theo nhóm LMK là đáp án đúng**

Nguyen Le Nhat Duong 1952638

Đỗ Đăng Khoa 1952295

Mai Tôn Nhật Khánh 1852038

Question 4:

Column major

1. A[4][3]
2. A: Array [-2..1,3..5] of real;

|  |
| --- |
| a[0][0] |
|  |
|  |
|  |
| a[1][0] |
|  |
|  |
|  |
| a[2][0] |
|  |
|  |
|  |
| a[3][0] |
|  |
|  |
|  |
| a[0][1] |
|  |
|  |
|  |

b)

A[3][3]

|  |
| --- |
| a[0][0] |
|  |
| a[1][0] |
|  |
| a[2][0] |
|  |
| a[0][1] |
|  |
| a[1][1] |
|  |

Row major:

1. a[4][3]

|  |
| --- |
| a[0][0] |
|  |
|  |
|  |
| a[0][1] |
|  |
|  |
|  |
| a[0][2] |
|  |
|  |
|  |
| a[1][0] |
|  |
|  |
|  |
| a[1][1] |
|  |
|  |
|  |

1. a[3][3]

|  |
| --- |
| a[0][0] |
|  |
| a[0][1] |
|  |
| a[0][2] |
|  |
| a[1][0] |
|  |
| a[1][1] |
|  |

Question 5:

1. A[x][y][z] size of element is SIZE

Row major:

Access A[x’][y’][z’] = Base Address + (x’\*y\*z + y’\*z + z’)\*SIZE

Column major:

Access A[x’][y’][z’] = Base Address + (x’\*y\*z + z’\*y + y’)\*SIZE

1. A[x1][x2]... [xn] size of element is SIZE

Row major

Access A[x1’][x2’]...[xn’] =

Base Address + (x1’ \* (x2x3...xn) + x2’\* (x3x4...xn) +...+x(n-1)’\*xn + xn’ )\*SIZE

Column major:

Access A[x1’][x2’]...[xn’] =

Base Address + (x1’ \* (x2x3...xn) + x2’\* (x3x4...xn) +...+ xn’\*x(n-1) + x(n-1)’ )\*SIZE

Question 6:

A: Integer => 2 bytes => padding 14 bytes

B: String[15] => 15 bytes + 1 byte for descriptor = 16 bytes

C: 32 bits => 4 bytes + 6 bytes padding = 10

Result = 1 bytes + 2 integer + 4 real = 7 we take the true case because its size is bigger

Padding 15 bytes more

=> Size of 16 + 16 + 10 + 22 = 64

b) We rearranged the record:

String b[15];

C: set of 1..32

Case result : ….

A: Integer

=> size 16 + 4 + 1 + 7 + 2 +2 = 32 bytes

Question 7:

f: T1->T2 (1)

T1 = (T3\*T4\*T5) (2)

1. And (2) => f: (T3\*T4\*T5)->T2 (3)

b(c) => T6->T7 (4)

1. And (4) => T6 = T5, T4 = T6-> T7

a = T3 = int (operands of + operator are int)

b(c) = T7 = int (operands of > must be of same type, a is int so b(c) must also be int)

The result is :

(int,(T->int),T)->int

Quach Dang Giang - 1952044

Bui Quang Tien - 1953018

Lương Duy Hưng - 1952747

**Question 4:**

**a)**

|  |  |  |
| --- | --- | --- |
| **Address** | **Row-major order** | **Column-major order** |
| **0** | **a[-2][3]** | **a[-2][3]** |
| **1** | **a[-2][4]** | **a[-1][3]** |
| **2** | **a[-2][5]** | **a[0][3]** |
| **3** | **a[-1][3]** | **a[1][3]** |
| **4** | **a[-1][4]** | **a[-2][4]** |
| **5** | **a[-1][5]** | **a[-1][4]** |

**b)**

|  |  |  |
| --- | --- | --- |
| **Address** | **Row-major order** | **Column-major order** |
| **0** | **a[-1][-2]** | **a[-1][-2]** |
| **1** | **a[-1][-1]** | **a[0][-2]** |
| **2** | **a[-1][0]** | **a[1][-2]** |
| **3** | **a[0][-2]** | **a[-1][-1]** |
| **4** | **a[0][-1]** | **a[0][-1]** |
| **5** | **a[0][0]** | **a[1][-1]** |

**Question 5:**

**a,**

**Row-major order:**

1. **3-dimension array**

Given: array[x][y][z] int x[5] => lower bound = 0

A [2 .. 6] of integer: lower bound x\_lb + upper bound x\_ub => x\_s = x\_ub - x\_lb + 1

with start memory location which is address of array[x\_lb, y\_lb, z\_lb] is **alpha**

E is the size of an element.

Location: a[i][j][k] = alpha + {[(i-x\_lb)\*y\_s + (j-y\_lb)]\*z\_s + (k-z\_lb)} \* E

= alpha + (i-x\_lb)\*y\_s \* z\_s \* E + (j-y\_lb)\*z\_s\* E + (k-z\_lb) \* E

1. **n-dimension array:**

Given: array[x1][x2]......[xn]:

with start memory location which is address of array[x\_lb, y\_lb, z\_lb…n\_lb] is **alpha**

location array[i1][i2]...[in]

alpha + [ ( (\_lb) \* ) ] \* E

**Column-major order:**

1. **3-dimension array**

Given: array[x][y][z]

with start memory location which is address of array[x\_lb, y\_lb, z\_lb] is **alpha**

E is the size of an element.

Then:

Location: a[i][j][k] = alpha + {[(i-x\_lb)\*y + (k-y\_lb)]\*z + (j-z\_lb)} \* E

1. **N-dimension array**

**Question 6:**

The size of the record given is 48 bytes (with data alligment)

That is

int = 2 => plus 14 bytes for padding

string is 16 => no padding

set is 4 bytes + result is 1 bytes => plus 6 bytes padding

Rearrange the component by putting variable with string type at the end/ at the beginning.

string is 16 => no padding

int is 2 bytes + set is 4 bytes + result is 6 bytes => plus 4 bytes padding

Then the size of record is now 32 bytes

a:integer

**Question 7:**

f(a,b,c) is function so its type is T1 -> T2

Function has 3 paras so T1 = T3\*T4\*T5, with T3 is type of a, T4 is type of b, T5 is type of c

There is expression b(c) in body, and b(c) and a has same type since we have a > b(c)

=> Type of function b is T4 = T5 -> T3, and b(c) has the type of T3

Function f return a+1 or b(c) => A is int, T3 = int, and return int

=> Type of f(a,b,c) is (int\*(T->int)\*T) -> int

Trần Quốc Việt - 1953097

Bui Hoang Phuc - 1952925

Pham Nhut Huy 1952059

**Question 4.**

I) Row-major

a)

|  |  |  |
| --- | --- | --- |
| A[-2,3] | A[-2,4] | A[-2,5] |
| A[-1,3] | A[-1,4] |  |

b)

|  |  |  |
| --- | --- | --- |
| A[-1,-2] | A[-1,-1] | A[-1,0] |
| A[0,-2] | A[0,-1] |  |

II) Column

a)

|  |  |
| --- | --- |
| A[-2,3] | A[-2,4] |
| A[-1,3] |  |
| A[0,3] |  |
| A[1,3] |  |

b)

|  |  |
| --- | --- |
| A[-1,-2] | A[-1,-1] |
| A[0,-2] | A[0,-1] |
| A[1,-2] |  |

**Question 5.**

3-Dimension:

Row: Location (a[i,j,k]) = a(row\_lb, col\_lb,height\_lb) + E\*( (i-row\_lb)\*n1\*n2 + (j-col\_lb)\*n2 + (k-height\_lb))

Column: Location (a[i,j,k]) = a(row\_lb, col\_lb,height\_lb) + E\*( (i-row\_lb) + (j-col\_lb)\*n1 + (k-height\_lb)\*n1\*n2)

N-dimension

Row: Location (a[i1,i2,...,ik]) = a(i1\_lb, i2\_lb,...,ik\_lb) + E\*( (i1-i1\_lb)\*n1\*...nk-1 + (i2-i2\_lb)\*n2\*...\*nk-1 + …. + (ik-ik\_lb))

Column: Location (a[i1,i2,...,ik]) = a(i1\_lb, i2\_lb,...,ik\_lb) + E\*( (i1-i1\_lb) + (i2-i2\_lb)\*n1+ …. + (ik-ik\_lb)\*n1\*..n\*n(k-1))

**Question 6:**

Integer: 2

String[15]: 1 + 15

Set: 32 bits = 4 bytes

Union: 1 + max((4+2),(2\*2)) = 7

In Q3: (2 + 2) + 16 + 4 + (7 + 1) = 32 bytes, where 2 and 1 byte(s) is for padding

In Q6, if align in following order: result (bool) → a (int) → b (string) → c(set)

Result: (7 + 2 + 3) + 16 + 4 = 32 bytes, where 3 bytes if for padding

→ There is no solution to rearrange the record to reduce the size.

**Cao Bá Huy - 1952713**

**Lê Minh Khôi - 1952076**

**Nguyễn Minh Tâm - 1952968**

**Huỳnh Minh Trí - 1953041**

### Question 4

1. Array[-2..1, 3..5]

Row-major order : A[-2,3] , A[-2,4] , A[-2,5] , A[-1,3] A[-1,4]

Column-major order: A[-2,3] , A[-1,3], A[0,3], A[1,3], A[-2,4]

1. Array[-1..1, -2..0]

Row-major order: A[-1, -2], A[-1, -1], A[-1, 0], A[0, -2], A[0, -1]

Column-major order: A[-1, -2], A[0, -2], A[1,-2], A[-1,-1] , A[0,-1].

### Question 5

1. Row-major order for a[ i ][ j ][ k ] in A[B][R][C] : Base + (( i - lowerBoundB) \* R\*C + (j - lowerBoundR )\*C + (k - lowerBoundC)) \* Size of 1 element.

Column-major order for a[ i ][ j ][ k ] in A[B][R][C] : Base + (( i - lowerBoundB) \* R\*C + (j - lowerBoundR ) + (k - lowerBoundC)\*R ) \* Size of 1 element.

1. Row-major order for a[ i1 ][ i2 ] …[ i{n-2} ] [ j ][ k ] in A[B1]...[B{n-2}][R][C] = Base + (( i1 - lowerBoundB1) \* B2 \* … \* B{n-2} \* R\*C + (i2 - lowerBoundB2)\* B3 \* … \* B{n-2} \* R\*C + …. + (j - lowerBoundR ) \* C + (k - lowerBoundC ) )\* Size 1 element

Column-major order for a[ i1 ][ i2 ] …[ i{n-2} ] [ j ][ k ] in A[B1]...[B{n-2}][R][C] = Base + (( i1 - lowerBoundB1) \* B2 \* … \* B{n-2} \* R\*C + (i2 - lowerBoundB2)\* B3 \* … \* B{n-2} \* R\*C + …. + (j - lowerBoundR ) + (k - lowerBoundC )\*R )\* Size 1 element

### Question 6

string (16 bytes) -> 16

set (4 bytes) -> 4

result (1 bytes) -> 1 + 3 padding

union (4 + 2 bytes) -> 6 -> Pad: 10 -> 16 bytes

integer (2 bytes)

→ **16 + 4 + 1+3 + 4 + 2 + 2= 32 bytes (Đáp án sau khi sắp xếp lại là 32 byte , Còn nếu chưa sắp xếp thì sẽ 48 byte -> Video demo nằm trong cùng folder)**

Order: string (16 bytes) -> 16

set (4 bytes) -> 4

integer (2 bytes) -> 2

result (1 bytes) -> 1

union (6 bytes) -> Pad: 10 -> 16

→ Sum: 16 + 4 + 2 + 1 + 16 = 39 bytes

### Question 7:

1. def f(a,b,c) => f is a 3-parameter function =>

f: (T1 \* T2 \* T3) -> T4 (1)

A: T1 (2)

B: T2 (3)

C: T3 (4)

1. b(c) => b is a 1-parameter function with argument c

B: T5-> T6 (5)

1. (3)& (5) => T2 = T5 -> T6 (6)
2. (4) &(5) => T3 = T5 (7)
3. a > b(c) & (2) & (5) => T1 = T6 (8)
4. a + 1 & (2) => T1 = integer (9)
5. (1) & def f(a,b,c) = … then a + 1 => return type of f = return type of a + 1 = integer

T4 = integer (10)

1. (5) & then a + 1 else b(c) => return type of b is integer => T6 = integer (11)
2. (1) … (11) => f: (integer \* (T3 \* integer) \* T3) -> integer

f: (integer \* (T \* integer) \* T) -> integer

typeQUIZ

T1 = T5 (if a > b(c) )

b: T3 -> T5

T1 + integer = T3 -> T5 = T4 (T2= T3->T5)

=> T1 = integer

=> T5 = integer

=> T4 = integer

f: integer \* (T3->integer) \* T3 ->integer

Phạm Bùi Minh Huân 1952056

Võ Ngọc Sang - 1952430

Phạm Khánh Trình - 1953044

**Question 4:**

a)

* Column major:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A[-2,3] | A[-1,3] | A[0,3] | A[1,3] | A[-2,4] |

* Row major:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A[-2,3] | A[-2,4] | A[-2,5] | A[-1,3] | A[-1,4] |

b)

* Column major:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A[-1,-2] | A[0,-2] | A[1,-2] | A[-1,-1] | A[0,-1] |

* Row major:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A[-1,-2] | A[-1,-1] | A[-1,0] | A[0,-2] | A[0,-1] |

**Question 5:**

1. Three dimensional array:

Column major:

arr(x1..x2, y1..y2, z1..z2)

**Address of[x][y][z] = [ (x – x1) + (y – y1) \* (x2-x1+1)\* (z2-z1+1) + (z – z1) ] \* size\_of\_ele + base\_address**

Row major:

(x1..x2, y1..y2, z1..z2)

**Address of[x][y][z] = [(x – x1) \* (y2-y1+1) + (y – y1)\* (z2-z1+1) + (z – z1)] \* size\_of\_ele + base\_address**

1. n dimensional array:

Column major:

arr[aa][aa+1]...[aa+n-1]: n dimensional array

in detail: (a1..a2, b1..b2, … , z1..z2)

**Address of[a][b]...[z] = base\_addess + size\_of\_ele \* [(a-a1) + (b-b1)\*aa\*(aa+2)\*...\*(aa+n-1) + (c-c1) + (d-d1) + … + (z-z1)]**

1952678-Đỗ Hoàng Hiếu

1952214 - Trần Quốc Duy

1952114 - Lâm Minh Quân



Static:

* Advantages: efficiency ( no dynamic allocation)
* Ex: static int myarray[3] = {0,1,2};

Fixed stack-dynamic:

* Advantages: space efficiency
* Ex: int array[3] = { 2,3,4};

Stack-dynamic:

* Advantages: flexibility (the size of an array need not be know until the array is to be used)
* Ex: cin »n; int x[n];

Fixed heap-dynamic:

* Advantages: flexibility
* Ex: int[] = new int[10];

Heap-dynamic:

* Advantages: arrays can grow or shrink during program execution.
* Ex: cin >> n; int[] x = new int[n];

2)

A floating point number type is stored based on IEEE 754 standard: Value of the number is represented by 9 bits, the first bit is sign bit (1 for negative and 0 for positive). The next three bits are used for exponent and the last five is for value.

(−1)sign × 1.mantissa × 2exponent−bias

bias = 23−1 − 1 = 310

a)

100101100:

sign bit is 1 -> negative

exp is 001 -> exp=1

fraction is 01100 -> fraction = 0.375

answer = (-1)^1 x (1+0.375) x 2^(1-3)= -0.34375

001010000:

sign bit is 0 -> positive

exp is 010 -> exp=2

fraction is 10000 -> fraction = 0.5

answer = (-1)^0 x (1+0.5) x 2^(2-3)= 0.75

111110000

sign bit is 1 -> negative

exp is 111 -> exp = 7

fraction is 10000 -> fraction = 0.0625

answer = (-1)^1 x (1+0.0625) x 2^(7-3) =

4)

1. A: Array [-2..1,3..5]

|  |  |
| --- | --- |
| Column major | Row major |
| A[-2,3] | A[-2,3] |
| A[-1,3] | A[-2,4] |
| A[0,3] | A[-1,3] |
| A[-2,4] | A[-1,4] |
| A[-1,4] | A[0,3] |

b) A: Array [-1..1,-2..0]

|  |  |
| --- | --- |
| Column major | Row major |
| [-1,-2] | A[-1,-2] |
| [0,-2] | A[-1,-1] |
| [-1,-1] | A[0,-2] |
| [0,-1] | A[0,-1] |
|  |  |

5)

a)

Array[a][b][c]

Row major:

Location(Array[i][j][k]) = Base address + ((i - a\_lb)\*b\*c + (j - b\_lb)\*c + (k - c\_lb))\*Ele\_size

`Column major:

Location(Array[i][j][k]) = Base address + ((k - c\_lb)\*b\*a + (j - b\_lb)\*a + (i - a\_lb))\*Ele\_size

b)

6)

7) def f(a,b,c) = if a > b(c) then a + 1 else b(c)

a+1: int type (because using operator “+”)

b(c): int type (then and else branches return the same type)

f(a,b,c): int type

Le Duc Cam - 1952588

Ho Tri Khang - 1952069

Huynh Phuoc Thien - 1952463

Nguyen Luat Gia Khoi

4.

1. Array[-2..1,3..5]

Row:





Column:









1. Array[-1..1,-2..0]

Row:





Column:







5)

1. Three-dimensional arrays

Row major: (x1..x2, y1..y2, z1..z2)

arr[i,j,k] = (((i-x1)\*(y2-y1+1) + (j-y1))\*(z2-z1+1) + (k-z1))\*element\_size + base\_address

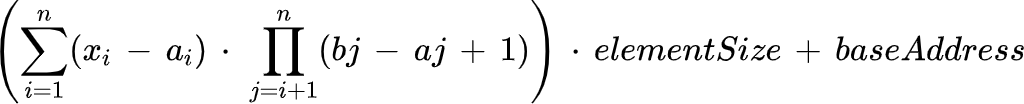
Column major:

arr[i,j,k] = (((i-x1) + (j-y1)\*(x2-x1+1))\*(z2-z1+1) + (k-z1))\*element\_size + base\_address

1. Multi-dimensional arrays:

Row major: (a1..b1, a2..b2, a3..b3, …, an..bn)

arr[x1, x2, …, xn] =

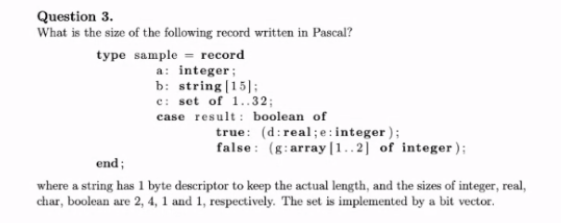


Column major:

arr[x1, x2, …, xn] =

{"code":"$$\\left(\\,\\left(\\left(x_{2}\\,-\\,a2\\,\\right)\\,\\cdot\\,\\left(b1-a1+1\\right)\\,+\\,\\left(x1\\,-\\,a1\\right)\\,\\right)\\cdot\\,\\prod_{i=3}^{n}\\left(bi\\,-\\,ai\\,+\\,1\\right)+\\left(\\sum_{i=3}^{n}\\left(x_{i}\\,-\\,a_{i}\\right)\\,\\cdot\\,\\prod_{j=i+1}^{n}\\left(bj\\,-\\,aj\\,+\\,1\\right)\\right)\\right)\\,\\cdot\\,elementSize\\,+\\,baseAddress$$","backgroundColorModified":false,"backgroundColor":"#ffffff","aid":null,"type":"$$","id":"1-1","font":{"size":11,"family":"Arial","color":"#000000"},"ts":1649120632953,"cs":"dZM+hxpuHQhUVkNYH+IiUw==","size":{"width":1020,"height":52}}

6.



integer: 2

real: 4

boolean: 1

string[15]: 16 (char 1)

set 1..32: 4

**Rearrange:**

b: string[15];

c: set of 1..32;

a: integer;

case result: boolean of

true: (d: real, e:integer);

false: (g: array[1..2] of integer)

16

4

2

1 + 1

4

2

=> Total size: 30