

花开一季 叶落一地 P2 课上

花开一季 叶落一地 P2 课上

冒泡排序

C语言（针对升序）

MIPS

拓展—选择排序C

初次快排（判断条件竟然不反过来写？亏你写的出来）

斐波那契数列

C代码

虚假的递归

真正的递归

数列求解

约瑟夫

递归+从k开始+有中间输出

迭代+ 从k开始+无中间输出

数组遍历

全排列

C代码

MIPS

矩阵乘法

高精度阶乘

C++

MIPS

冒泡排序

C语言（针对升序）

```

1 void bubbleSort(int k[], int n)
2 {
3     int i, j, temp, flag;
4     for (i = 0; i < n - 1; i++)
5     {
6         for (j = 0, flag = 0; j < n - i - 1; j++)
7             if (k[j] > k[j + 1])
8                 { /*这里是升序，如果是降序都会改吧*/
9                     temp = k[j];
10                    k[j] = k[j + 1];
11                    k[j + 1] = temp; /* 交换两个元素的位置 */
12                    flag = 1;
13                }
14     if (flag == 0)

```

```

15         break;
16     }
17 }

```

MIPS

```

1  .data
2  ans: .space 4080
3  space: .asciiz " "
4
5  .macro read(%n)
6      li $v0, 5
7      syscall
8      move %n, $v0
9  .end_macro
10
11 .macro write(%n, %h)
12     move $a0, %n
13     li $v0, %h
14     syscall
15 .end_macro
16
17 .text
18 read($s0) # $s0 = n
19 li $s1, 0 # $s1 = i
20 li $s2, 0 # $s2 = 4 * i
21
22 input:
23     bge $s1, $s0, sort
24     read($s3)
25     sw $s3, ans($s2)
26     addi $s1, $s1, 1
27     addi $s2, $s2, 4
28     j input
29
30 sort:
31     li $s1, 0
32     subi $s3, $s0, 1 # $s3 = n - 1
33
34     for_i:
35         bge $s1, $s3, end_i
36         li $s4, 0 # $s4 = flag
37         li $s5, 0 # $s5 = j
38         sub $s7, $s3, $s1 # $s7 = n - 1 - i

```

```

39
40         for_j:
41             bge $s5, $s7, end_j
42             sll $t2, $s5, 2
43             addi $t3, $t2, 4
44             lw $t0, ans($t2)
45             lw $t1, ans($t3)
46             ble $t0, $t1, end_swap
47             sw $t0, ans($t3)
48             sw $t1, ans($t2)
49             li $s4, 1
50
51     end_swap:
52         addi $s5, $s5, 1
53         j for_j
54
55     end_j:
56         beqz $s4, end_i
57         addi $s1, $s1, 1
58         j for_i
59
60 end_i:
61     li $s1, 0 # $s1 = i
62     li $s2, 0 # $s2 = 4 * i
63
64 output:
65     bge $s1, $s0, end
66     lw $s3, ans($s2)
67     write($s3, 1)
68     la $s4, space
69     write($s4, 4)
70     addi $s1, $s1, 1
71     addi $s2, $s2, 4
72     j output
73
74 end:
75     li $v0, 10
76     syscall

```

拓展—选择排序C

```

1 void selectSort(int k[], int n)
2 {
3     int i, j, d, temp;
4     for (i = 0; i < n - 1; i++)
5     {
6         d = i;
7         for (j = i + 1; j < n; j++)
8             if (k[j] < k[d])
9                 d = j;
10        if (d != i)
11        {
12            temp = k[d]; /* 最小值元素非未排序元素的第一个元素时 */
13            k[d] = k[i];
14            k[i] = temp;
15        }
16    }
17 }

```

初次快排（判断条件竟然不反过来写？亏你写的出来）

```

1 .data
2     ans: .space 4096
3     space: .asciiz ","
4     left: .asciiz "["
5     right: .asciiz "]"
6 .text
7     li $v0, 5
8     syscall
9     move $t0, $v0 # t0, n
10
11     la $s0, ans # s0, ans
12     move $t1, $zero # t1, i
13
14 input:
15     beq $t1, $t0, sort
16     li $v0, 5
17     syscall
18     sll $t2, $t1, 2 # t2 i * 4
19     add $t2, $s0, $t2
20     sw $v0, 0($t2)
21     addi $t1, $t1, 1
22     j input

```

```

23
24 sort:
25     move $a0, $zero # a0 i
26     subi $a1, $t0, 1 # a1 j
27     jal quicksort
28     move $t1, $zero
29
30     la $a0, left
31     li $v0, 4
32     syscall
33
34     j print
35
36 quicksort:
37     blt $a0, $a1, owo
38 owon:
39     jr $ra
40
41 owo:
42
43     move $t5, $a0
44     move $t6, $a1
45     sll $t2, $t5, 2
46     add $t2, $s0, $t2
47     lw $t4, 0($t2) # t4 key
48
49 while1:
50     bge $t5, $t6, get2
51
52 while2:
53     blt $t5, $t6, con1
54     j while1
55
56 con1:
57     sll $t2, $t6, 2
58     add $t2, $s0, $t2
59     lw $t9, 0($t2)
60
61     bge $t9, $t4, con2
62
63     sll $t2, $t5, 2
64     add $t2, $s0, $t2
65     sw $t9, 0($t2)
66
67     j while3

```

```

68
69 con2:
70     subi $t6, $t6, 1
71     j while2
72
73 while3:
74     blt $t5, $t6, con3
75     j while1
76
77
78 con3:
79     sll $t2, $t5, 2
80     add $t2, $s0, $t2
81     lw $t8, 0($t2)
82
83     ble $t8, $t4, con4
84
85     sll $t2, $t6, 2
86     add $t2, $s0, $t2
87     sw $t8, 0($t2)
88
89     j while1
90
91 con4:
92     addi $t5, $t5, 1
93     j while3
94
95 get2:
96     sll $t2, $t5, 2
97     add $t2, $s0, $t2
98     sw $t4, 0($t2)
99
100    move $v0, $t5
101
102    subi $sp, $sp, 32
103    sw $ra, 0($sp)
104    sw $a0, 4($sp)
105    sw $a1, 8($sp)
106    sw $v0, 12($sp)
107    subi $a1, $v0, 1
108    jal quicksort
109    lw $ra, 0($sp)
110    lw $a0, 4($sp)
111    lw $a1, 8($sp)
112    lw $v0, 12($sp)

```

```

113     addi $sp, $sp, 32
114
115     subi $sp, $sp, 32
116     sw $ra, 0($sp)
117     sw $a0, 4($sp)
118     sw $a1, 8($sp)
119     sw $v0, 12($sp)
120     addi $a0, $v0, 1
121     jal quicksort
122     lw $ra, 0($sp)
123     lw $a0, 4($sp)
124     lw $a1, 8($sp)
125     lw $v0, 12($sp)
126     addi $sp, $sp, 32
127     j owon
128
129     print:
130     beq $t1, $t0, exit
131     sll $t2, $t1, 2
132     add $t2, $s0, $t2
133     lw $a0, 0($t2)
134     li $v0, 1
135     syscall
136     addi $t1, $t1, 1
137     beq $t1, $t0, exit
138     la $a0, space
139     li $v0, 4
140     syscall
141     j print
142
143
144     exit:
145     la $a0, right
146     li $v0, 4
147     syscall
148     li $v0, 10
149     syscall

```

斐波那契数列

C代码

```

1  int f(int n)
2  {
3      if (n == 0)
4          return 0;
5      if (n == 1)
6          return 1;
7      return f(n - 1) + f(n - 2);
8  }
9
10 int main()
11 {
12     int n;
13     scanf("%d", &n);
14     printf("%d", f(n));
15     return 0;
16 }

```

虚假的递归

```

1  .data
2
3  .macro read(%n)
4      li $v0, 5
5      syscall
6      move %n, $v0
7  .end_macro
8
9  .macro write(%n)
10     move $a0, %n
11     li $v0, 1
12     syscall
13 .end_macro
14
15 .text
16 read($s0) # $s0 = n
17 move $a0, $s0 # $a0 = n
18 li $s1, 0 # $s1 = ans
19 jal f
20 write($s1)
21 li $v0, 10
22 syscall
23

```



```

24 f:
25     beq $a0, 0, f0
26     beq $a0, 1, f1
27     subi $sp, $sp, 8
28     sw $a0, 0($sp)
29     sw $ra, 4($sp)
30     subi $a0, $a0, 1
31     jal f
32     subi $a0, $a0, 1
33     jal f
34     lw $a0, 0($sp)
35     lw $ra, 4($sp)
36     addi $sp, $sp, 8
37     jr $ra
38
39 f0:
40     addi $s1, $s1, 0
41     jr $ra
42
43 f1:
44     addi $s1, $s1, 1
45     jr $ra

```

真正的递归

```

1  .data
2
3  .macro read(%n)
4      li $v0, 5
5      syscall
6      move %n, $v0
7  .end_macro
8
9  .macro write(%n)
10     move $a0, %n
11     li $v0, 1
12     syscall
13 .end_macro
14
15 .text
16 read($s0) # $s0 = n
17 move $a0, $s0 # $a0 = n
18 li $s1, 0 # $s1 = ans
19 jal f

```

```

20 write($v0)
21 li $v0, 10
22 syscall
23
24 f:
25     beq $a0, 0, f0
26     beq $a0, 1, f1
27     subi $sp, $sp, 8
28     sw $a0, 0($sp)
29     sw $ra, 4($sp)
30     subi $a0, $a0, 1
31     jal f
32     subi $sp, $sp, 4
33     sw $v0, 0($sp)
34     subi $a0, $a0, 1
35     jal f
36     lw $s1, 0($sp)
37     addi $sp, $sp, 4
38     move $s2, $v0
39     add $v0, $s1, $s2 # f(n) = f(n - 1) + f(n - 2)
40     lw $a0, 0($sp)
41     lw $ra, 4($sp)
42     addi $sp, $sp, 8
43     jr $ra
44
45 f0:
46     li $v0, 0
47     jr $ra
48
49 f1:
50     li $v0, 1
51     jr $ra

```

数列求解

```

1 .data
2     ans: .space 4080
3
4 .macro read(%n)
5     li $v0, 5
6     syscall
7     move %n, $v0
8 .end_macro
9

```

```

10 .macro write(%n)
11     move $a0, %n
12     li $v0, 1
13     syscall
14 .end_macro
15
16 .text
17 read($s0) # $s0 = n
18 li $s1, 2 # $s1 = i
19 li $s2, 0 # $s2 = i - 2
20 li $s3, 1 # $s1 = i - 1
21 sw $s2, ans
22 sw $s3, ans+4
23
24 loop:
25     bgt $s1, $s0, end
26     sll $s1, $s1, 2
27     sll $s2, $s2, 2
28     sll $s3, $s3, 2
29     lw $s4, ans($s2)
30     lw $s5, ans($s3)
31     add $s6, $s4, $s5
32     sw $s6, ans($s1)
33     srl $s1, $s1, 2
34     srl $s2, $s2, 2
35     srl $s3, $s3, 2
36     addi $s1, $s1, 1
37     addi $s2, $s2, 1
38     addi $s3, $s3, 1
39     j loop
40
41 end:
42     sll $s0, $s0, 2
43     lw $s1, ans($s0)
44     write($s1)
45     li $v0, 10
46     syscall

```

约瑟夫

递归+从k开始+有中间输出

- C

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  //用递归实现约瑟夫环问题
4
5  int cir(int n, int m, int i)
6  {
7      if (i == 1)
8          return (m - 1 + n) % n;
9      return (cir(n - 1, m, i - 1) + m) % n;
10 }
11 int main()
12 {
13     int n, m, k;
14     cin >> n >> m >> k;
15     for (int i = 1; i <= n; i++)
16     {
17         int tmp = (cir(n, m, i) + k) % n;
18         if (tmp == 0)
19             printf("%d\n", n);
20         else
21             printf("%d\n", tmp);
22     }
23     return 0;
24 }

```

- MIPS(zes)

```

1  .data
2      Enter:.asciiz "\n"
3  .macro read(%n)
4      li $v0, 5
5      syscall
6      move %n, $v0
7  .end_macro
8
9  .macro write(%n)
10     move $a0, %n
11     li $v0, 1
12     syscall
13 .end_macro
14

```

```

15 .text
16     read($s0) #s0-n
17     read($s1) #s1-m
18     read($s7) #s7-k
19
20     li $t0, 1 #t0-1
21     li $s2, 0 #s2-ans
22 For:
23     bgt $t0, $s0, End
24     move $a0, $s0 #n
25     move $a1, $s1 #m
26     move $a2, $t0 #i
27     jal cir
28     add $s2, $s2, $s7 #ans = (cir(n,m,i) + k)
29     div $s2, $s0 #ans = (cir(n,m,i) + k) % n
30     mfhi $s2
31     beq $s2, $0, isZero
32     write($s2) #ans = (cir(n,m,i) + k) % n
33     j continue
34 isZero:
35     write($s0) #ans = n
36 continue:
37     la $a0, Enter
38     li $v0, 4
39     syscall
40     li $s2, 0
41
42     addi $t0,$t0, 1
43     j For
44 cir:
45     addi $sp, $sp, -16
46     sw $ra, 0($sp)
47     sw $a0, 4($sp)
48     sw $a1, 8($sp)
49     sw $a2, 12($sp)
50
51     beq $a2, 1, i_1
52     addi $a0, $a0, -1
53     addi $a2, $a2, -1
54     jal cir
55
56     lw $ra, 0($sp)
57     lw $a0, 4($sp)
58     lw $a1, 8($sp)
59     lw $a2, 12($sp)

```

```

60    addi $sp, $sp, 16
61    add $s2, $s2, $a1
62    div $s2, $a0
63    mfhi $s2
64    jr $ra
65    i_1:
66    add $s2, $s2, $a1
67    addi $s2, $s2, -1
68    add $s2, $s2, $a0
69    div $s2, $a0
70    mfhi $s2
71    lw $ra, 0($sp)
72    lw $a0, 4($sp)
73    lw $a1, 8($sp)
74    lw $a2, 12($sp)
75    addi $sp, $sp, 16
76    jr $ra
77    End:
78    li $v0, 10
79    syscall

```

迭代+ 从k开始+无中间输出

- C

```

1  #include <stdio.h>
2  int n, m, k;
3  int cir(int n, int m)
4  {
5      int p = 0;
6      int i;
7      for (i = 2; i <= n; i++)
8          p = (p + m) % i;
9      return p + 1;
10 }
11 int main()
12 {
13     while (~scanf("%d%d%d", &n, &m, &k))
14         printf("%d\n", (cir(n, m) + k) % n == 0 ? n : (cir(n, m) + k) % n);
15 }

```

- MIPS(lxy)

```

1  .macro read_int(%n)

```

```

2    li $v0, 5
3    syscall
4    move %n, $v0
5    .end_macro
6
7    .macro write_int(%n)
8        move $a0, %n
9        li $v0, 1
10       syscall
11    .end_macro
12
13    .text
14    main:
15        read_int($t0) # n
16        read_int($t1) # m
17        read_int($t2) # k
18
19        move $a0, $t0
20        move $a1, $t1
21        move $a2, $t2
22
23        jal ysf
24
25        move $t0, $a0
26        write_int($t0)
27
28        li $v0, 10
29        syscall
30
31    ysf:
32        move $t0, $a0 # n
33        move $t1, $a1 # m
34        move $t4, $a2 # k
35
36        li $t2, 0 # ans = 0
37        li $t3, 2 # i = 2
38
39        loop:
40        bgt $t3, $t0, end_loop
41        add $t2, $t2, $t1 #ans = ans + m
42        div $t2, $t3
43        mfhi $t2
44        addi $t3, $t3, 1
45        j loop
46

```

```

47  end_loop:
48  add $t2, $t2, $t4  #ans = ans + x
49  div $t2, $t0  #ans + x % n
50  mfhi $t2
51  beqz $t2, moven  # ans ? ans : n
52  move $a0, $t2
53  j return
54  moven:
55      move $a0, $t0
56  return:
57  jr $ra

```

数组遍历

- C (lh)

```

1  #include <iostream>
2  using namespace std;
3  bool visit[200];
4  int main()
5  {
6      int m, n;
7      cin >> m >> n;
8      int s = 0, num = 0;
9      for (int i = 1; num < m; i++)
10     {
11         if (i > m)
12             i %= m;
13         if (visit[i])
14             continue;
15         s++;
16         if (s == n)
17         {
18             cout << i << " ";
19             visit[i] = true;
20             s = 0;
21             num++;
22         }
23     }
24     return 0;
25 }

```

- MIPS (lh)


```

1  .data
2      array:.space 4000
3      b:.asciiz" "
4  .text
5  Input:
6      li $v0,5
7      syscall
8      move $s0,$v0#m个人
9      li $v0,5
10     syscall
11     move $s1,$v0#n为所报数字
12     li $t0,0
13     li $t4,1#i
14     li $t5,0#s
15     li $t6,0#num
16     li $s2,1#与1比较
17  Init:
18     beq $t0,$s0,For
19     move $t1,$t0
20     sll $t1,$t1,2
21     sw $0,array($t1)
22     addi $t0,$t0,1
23     j Init
24  For:
25     bge $t6,$s0,END
26     bgt $t4,$s0,DIV
27  After_DIV:
28     move $t1,$t4
29     sll $t1,$t1,2
30     lw $t2,array($t1)
31     beq $t2,$s2,continue
32     addi $t5,$t5,1
33     beq $t5,$s1,Print
34     addi $t4,$t4,1
35     j For
36  DIV:
37     div $t4,$s0
38     mfhi $t4
39     j After_DIV
40  continue:
41     addi $t4,$t4,1
42     j For
43  Print:
44     li $v0,1
45     move $a0,$t4

```

```

46    syscall
47    li $v0,4
48    la $a0,b
49    syscall
50    #visited[i] = true
51    move $t1,$t4
52    sll $t1,$t1,2
53    sw $s2,array($t1)
54    li $t5,0
55    addi $t6,$t6,1
56    addi $t4,$t4,1
57    j For
58 END:
59    li $v0,10
60    syscall

```

- MIPS(lsr)

```

1  .data
2  n: .word 0
3  m: .word 0
4  a: .space 1024
5  char: .asciiz "\n"
6  .text
7  la $t0, n
8  la $t1, m
9  la $s0, a
10 li $t2, 0#cnt
11 li $t3, 0#i
12 li $t4, 0#k
13 #输入n
14 li $v0, 5
15 syscall
16 sw $v0, ($t7)
17 lw $t0, ($t7)
18 #输入m
19 li $v0, 5
20 syscall
21 sw $v0, ($t7)
22 lw $t1, ($t7)
23 #跳转进入循环
24 jal choose
25
26 choose:

```

```

27 beq $t2, $t0, end#所有人均出局
28 #cnt!=n
29 addi $t3, $t3, 1#i+=1
30 bgt $t3, $t0, restart#判断i>n?
31 jal if_change
32
33 restart:
34 li $t3, 1
35 jal if_change
36
37 if_change:
38 #t6=a[i]
39 mul $t5, $t3, 4
40 add $t5, $t5, $s0
41 lw $t6, 0($t5)
42
43 beq $t6, 0, change#a[i]==0?
44 jal choose
45
46 change:
47 addi $t4, $t4, 1
48 beq $t4, $t1, change_#if k==m?
49 jal choose#k!=m
50
51 change_:
52 #t6=a[i]
53 mul $t5, $t3, 4
54 add $t5, $t5, $s0
55 lw $t6, 0($t5)
56 #a[i]=1
57 li $t8, 1
58 sw $t8, 0($t5)
59
60 addi $t2, $t2, 1#cnt+=1
61 #输出
62 li $v0, 1
63 move $a0, $t3
64 syscall
65 li $v0, 4
66 la $a0, char
67 syscall
68
69 li $t4, 0#k=0
70 jal choose
71

```

```

72 end:
73 li $v0, 10
74 syscall

```

全排列

C代码

```

1  #include <stdio.h>
2  #define MAXL (25)
3
4  int symbol[MAXL], array[MAXL], n;
5
6  void FullArray(int index)
7  {
8      int i;
9      if (index >= n)
10     {
11         for (i = 0; i < n; i++)
12             printf("%d ", array[i]);
13         putchar('\n');
14         return;
15     }
16     for (i = 0; i < n; i++)
17         if (symbol[i] == 0)
18         {
19             array[index] = i + 1;
20             symbol[i] = 1;
21             FullArray(index + 1);
22             symbol[i] = 0;
23         }
24 }
25
26 int main()
27 {
28     scanf("%d", &n);
29     FullArray(0);
30     return 0;
31 }

```

MIPS

```

1  .data
2  symbol: .space 100
3  array: .space 100
4  space: .asciiz " "
5  enter: .asciiz "\n"
6
7  .macro read(%n)
8      li $v0, 5
9      syscall
10     move %n, $v0
11 .end_macro
12
13 .macro write(%n, %h)
14     move $a0, %n
15     li $v0, %h
16     syscall
17 .end_macro
18
19 .text
20 read($s0) # $s0 = n
21 li $a0, 0 # $a0 = index
22 jal dfs
23 li $v0, 10
24 syscall
25
26 dfs:
27     blt $a0, $s0, no_print
28     li $t0, 0 # $t0 = i
29
30 print:
31     bge $t0, $s0, print_enter
32     sll $t1, $t0, 2
33     lw $t2, array($t1)
34     write($t2, 1)
35     la $t2, space
36     write($t2, 4)
37     addi $t0, $t0, 1
38     j print
39
40 print_enter:
41     la $t2, enter
42     write($t2, 4)
43     jr $ra

```

```

44
45 no_print:
46     li $t0, 0
47     loop:
48         bge $t0, $s0, return
49         sll $t1, $t0, 2
50         lw $t2, symbol($t1)
51         bnez $t2, end_if
52         addi $t3, $t0, 1
53         sll $t4, $a0, 2
54         sw $t3, array($t4)
55         li $t4, 1
56         sw $t4, symbol($t1)
57
58         subi $sp, $sp, 12
59         sw $a0, 0($sp)
60         sw $t0, 4($sp)
61         sw $ra, 8($sp)
62         addi $a0, $a0, 1
63         jal dfs
64         lw $a0, 0($sp)
65         lw $t0, 4($sp)
66         lw $ra, 8($sp)
67         addi $sp, $sp, 12
68
69         sll $t1, $t0, 2
70         li $t2, 0
71         sw $t2, symbol($t1)
72
73 end_if:
74     addi $t0, $t0, 1
75     j loop
76
77 return:
78     jr $ra

```

矩阵乘法

```

1  .data
2  a: .space 256
3  B: .space 256
4  c: .space 256
5
6  str_enter: .asciiz "\n"

```

```

7  str_space: .asciiz " "
8
9  .macro RI(%i)
10     li $v0, 5
11     syscall
12     move %i, $v0
13 .end_macro
14
15 .macro PI(%i)
16     li $v0, 1
17     move $a0, %i
18     syscall
19 .end_macro
20
21 .macro Pstr(%i)
22     la $a0, %i
23     li $v0, 4
24     syscall
25 .end_macro
26
27 .macro getIndex(%ans, %i, %j)
28     sll %ans, %i, 3
29     add %ans, %ans, %j
30     sll %ans, %ans, 2
31 .end_macro
32
33 .macro LOAD_LOCAL(%i)
34     addi $sp, $sp, 4
35     lw %i, 0($sp)
36 .end_macro
37
38 .macro SAVE_LOCAL(%i)
39     sw $i, 0($sp)
40     subi $sp, $sp, 4
41 .end_macro
42
43 .text
44     RI($s0)
45
46     li $t0, 0
47 in_1_i:
48     beq $t0, $s0, in_1_i_end
49     li $t1, 0
50 in_1_j:
51     beq $t1, $s0, in_1_j_end

```

```

52     RI($v0)
53     getIndex($t2, $t0, $t1)
54     sw $v0, a($t2)
55     addi $t1, $t1, 1
56     j in_1_j
57 in_1_j_end:
58     addi $t0, $t0, 1
59     j in_1_i
60 in_1_i_end:
61
62     li $t0, 0
63 in_2_i:
64     beq $t0, $s0, in_2_i_end
65     li $t1, 0
66 in_2_j:
67     beq $t1, $s0, in_2_j_end
68     RI($v0)
69     getIndex($t2, $t0, $t1)
70     sw $v0, B($t2)
71     addi $t1, $t1, 1
72     j in_2_j
73 in_2_j_end:
74     addi $t0, $t0, 1
75     j in_2_i
76 in_2_i_end:
77
78     li $t0, 0 # i
79 for_i:
80     beq $t0, $s0, for_i_end
81     li $t1, 0 # j
82 for_j:
83     beq $t1, $s0, for_j_end
84     li $t2, 0 # k
85     li $t3, 0 ##c[i][j]
86 for_k:
87     beq $t2, $s0, for_k_end
88     getIndex($t4, $t0, $t2)
89     lw $t6, a($t4)
90     getIndex($t5, $t2, $t1)
91     lw $t7, B($t5)
92     mul $t4, $t6, $t7
93     add $t3, $t3, $t4
94     add $t2, $t2, 1
95     j for_k
96 for_k_end:

```



```

97     getIndex($t4, $t0, $t1)
98     sw $t3, c($t4)
99     addi $t1, $t1, 1
100    j for_j
101    for_j_end:
102    addi $t0, $t0, 1
103    j for_i
104    for_i_end:
105
106    li $t0, 0
107    out_i:
108    beq $t0, $s0, out_i_end
109
110    li $t1, 0
111    out_j:
112    beq $t1, $s0, out_j_end
113    getIndex($t2, $t0, $t1)
114    lw $t3, c($t2)
115    PI($t3)
116    Pstr(str_space)
117    addi $t1, $t1, 1
118    j out_j
119    out_j_end:
120
121    Pstr(str_enter)
122    addi $t0, $t0, 1
123    j out_i
124    out_i_end:
125
126    li $v0, 10
127    syscall

```

高精度阶乘

C++

```

1  #include <iostream>
2  using namespace std;
3  const int length = 2500; //这个值经过多次调整，才过了10000!
4  int a[length];
5  int main()
6  {
7      a[0] = 1; //首位置为1;
8      int n;

```

```

9      cin >> n; //输入要计算阶乘的数
10     for (int i = 2; i <= n; i++)
11     {
12         int jinwei = 0;
13         int j = 0;
14         int temp;
15         while (j < length)
16         {
17             temp = a[j] * i + jinwei;
18             jinwei = temp / 10;
19             a[j] = temp % 10;
20             j++;
21         }
22     }
23     int k = length - 1;
24     while (!a[k])
25     { //将为0的数全跳过，不输出
26         k--;
27     }
28     // printf("%d\n", k);
29     while (k >= 0)
30     { //输出正确的阶乘结果
31         cout << a[k];
32         k--;
33     }
34     return 0;
35 }

```

MIPS

```

1  .data
2  a: .word 0:2500
3
4  .macro RI(%i)
5      li $v0, 5
6      syscall
7      move %i, $v0
8  .end_macro
9
10 .macro PI(%i)
11     li $v0, 1
12     move $a0, %i
13     syscall
14 .end_macro

```

```

15
16 .macro getIndex(%ans, %i)
17     sll %ans, %i, 2
18 .end_macro
19
20 .text
21 main:
22     RI($s0)
23     li $s1, 1 # len
24     li $t6, 10
25     li $t0, 1
26     sw $t0, a($0)
27     addi $t0, $t0, 1 # i
28
29     for:
30         bgt $t0, $s0, endfor
31         li $t2, 0 # carry
32         li $t1, 0 # j
33         while:
34             bge $t1, $s1, endwhile
35             getIndex($t4, $t1)
36             lw $t5, a($t4)
37             mul $t5, $t5, $t0
38             add $t5, $t5, $t2
39             div $t5, $t6
40             mflo $t2
41             mfhi $t5
42             sw $t5, a($t4)
43             addi $t1, $t1, 1
44             j while
45         endwhile:
46
47         whileC:
48             beqz $t2, endwhileC
49             div $t2, $t6
50             mflo $t2
51             mfhi $t5
52             getIndex($t4, $s1)
53             sw $t5, a($t4)
54             addi $s1, $s1, 1
55             j whileC
56         endwhileC:
57
58         addi $t0, $t0, 1
59         j for

```

```

60     endfor:
61
62     move $t0, $s1
63     while0:
64         getIndex($t1, $t0)
65         lw $t2, a($t1)
66         bgtz $t2, endwhile0
67         subi $t0, $t0, 1
68         j while0
69     endwhile0:
70
71     whileOut:
72         bltz $t0, endwhileOut
73         getIndex($t1, $t0)
74         lw $t2, a($t1)
75         PI($t2)
76         subi $t0, $t0, 1
77         j whileOut
78     endwhileOut:
79
80     li $v0, 10
81     syscall

```