ໂຄງສ້າງຂໍ້ມູນ ແລະ ຂັ້ນຕອນວິທີ (Data Structure and Algorithm)

ອາຈານສອນ: ບົວສິດ ໄຊຍະຈັກ

ວຸດທິການສຶກສາ: ປະລິຍາໂທວິທະຍາສາດ (ວິທະຍາສາດຄອມພິວເຕີ)

ຕຳແໜ່ງບໍລິຫານ: ຫົວໜ້າພະແນກການເງິນຂັ້ນສອງ

ໂທລະສັບມືຖື: (+856-20) 22245134

ອີເມລ: bouasoth@nuol.edu.la

ບົດທີ 1

ຄວາມຮູ້ເບື້ອງຕົ້ນກ່ຽວກັບໂຄງສ້າງຂໍ້ມູນ ແລະຂັ້ນຕອນວິທີ

ເນື້ອໃນຫຍໍ້

- ນິຍາມຂອງໂຄງສ້າງຂໍ້ມູນ ແລະ ຂັ້ນຕອນວິທີ
- ວິທີສ້າງ Algorithm
- ພື້ນຖານທາງຄະນິດສາດ
- Abstract Data Type (ADT)
- ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array
- Recursive Function

ນິຍາມໂຄງສ້າງຂໍ້ມູນ ແລະ ລຳດັບຂັ້ນຕອນຂອງຄຳສັ່ງ

- ໂຄງສ້າງຂໍ້ມູນ (Data Structure)
  - ໝາຍເຖິງການລວບລວມເອົາປະເພດຂໍ້ມູນມາໄວ້ຮ່ວມກັນຊຶ່ງເອີ້ນວ່າ ກຸ່ມປະເພດຂອງຂໍ້ມູນທີ່ມີການກຳນົດຄວາມສຳພັນກັນພາຍໃນກຸ່ມຂອງຂໍ້ມູນໄວ້ຢ່າງຈະແຈ້ງ ເພື່ອເອົາມາປະຍຸກໃຊ້ໃນການຂຽນໂປຣແກຣມ ຊຶ່ງການລວມກຸ່ມນັ້ນອາດຈະເປັນການລວມກຸ່ມກັນລະຫວ່າງຂໍ້ມູນປະເພດດຽວກັນ, ຕ່າງປະເພດ ຫຼື ຕ່າງໂຄງສ້າງຂໍ້ມູນກໍ່ໄດ້
  - ສ່ວນຄວາມສຳພັນພາຍໃນກຸ່ມຂອງຂໍ້ມູນນັ້ນແມ່ນບັນດາຫລັກການ, ຂໍ້ບັງຄັບຕ່າງໆທີ່ໃຊ້ຜູກມັດຂໍ້ມູນໄວ້ຮ່ວມກັນຢ່າງເປັນລະບົບ

ນິຍາມໂຄງສ້າງຂໍ້ມູນ ແລະ ລຳດັບຂັ້ນຕອນຂອງຄຳສັ່ງ

- ລຳດັບຂັ້ນຕອນຂອງຄຳສັ່ງ (Algorithm)
  - ແມ່ນວິທີໃນການຈັດລຽງລຳດັບຄຳສັ່ງໃນໂປຣແກຣມໃດໜຶ່ງ ເພື່ອໃຫ້ຄອມພິວເຕີປະຕິບັດງານຕາມຂັ້ນຕອນດັ່ງກ່າວ
    ຫລື
  - ແມ່ນລຳດັບຂັ້ນຕອນວິທີໃນການເຮັດວຽກຂອງໂປຣແກຣມເພື່ອແກ້ໄຂບັນຫາໃດໜຶ່ງ

ວິທີສ້າງ Algorithm

- ລຳດັບຂັ້ນຕອນຂອງຄຳສັ່ງ (Algorithm)
  - ການຂຽນ Algorithm ສາມາດຂຽນໄດ້ຫລາຍແບບເຊັ່ນ
    - Flow Chart
    - Pseudo Code
    - ພາສາມະນຸດ
  - ສາມາດຂຽນ Algorithm ເພື່ອແກ້ໄຂບັນຫາໜຶ່ງໄດ້ຫລາຍແບບ
    - ແຕ່ລະແບບ ຄອມພິວເຕີຈະໃຊ້ໜ່ວຍຄວາມຈຳ ແລະ ເວລາໃນການປະມວນຜົນແຕກຕ່າງກັນ
    - ສະນັ້ນ ຖ້າຕ້ອງການປຽບທຽບວ່າໂປຣແກຣມໃດດີກໍ່ຕ້ອງເບິ່ງຈາກປະສິດທິພາບຂອງ Algorithm ນັ້ນ

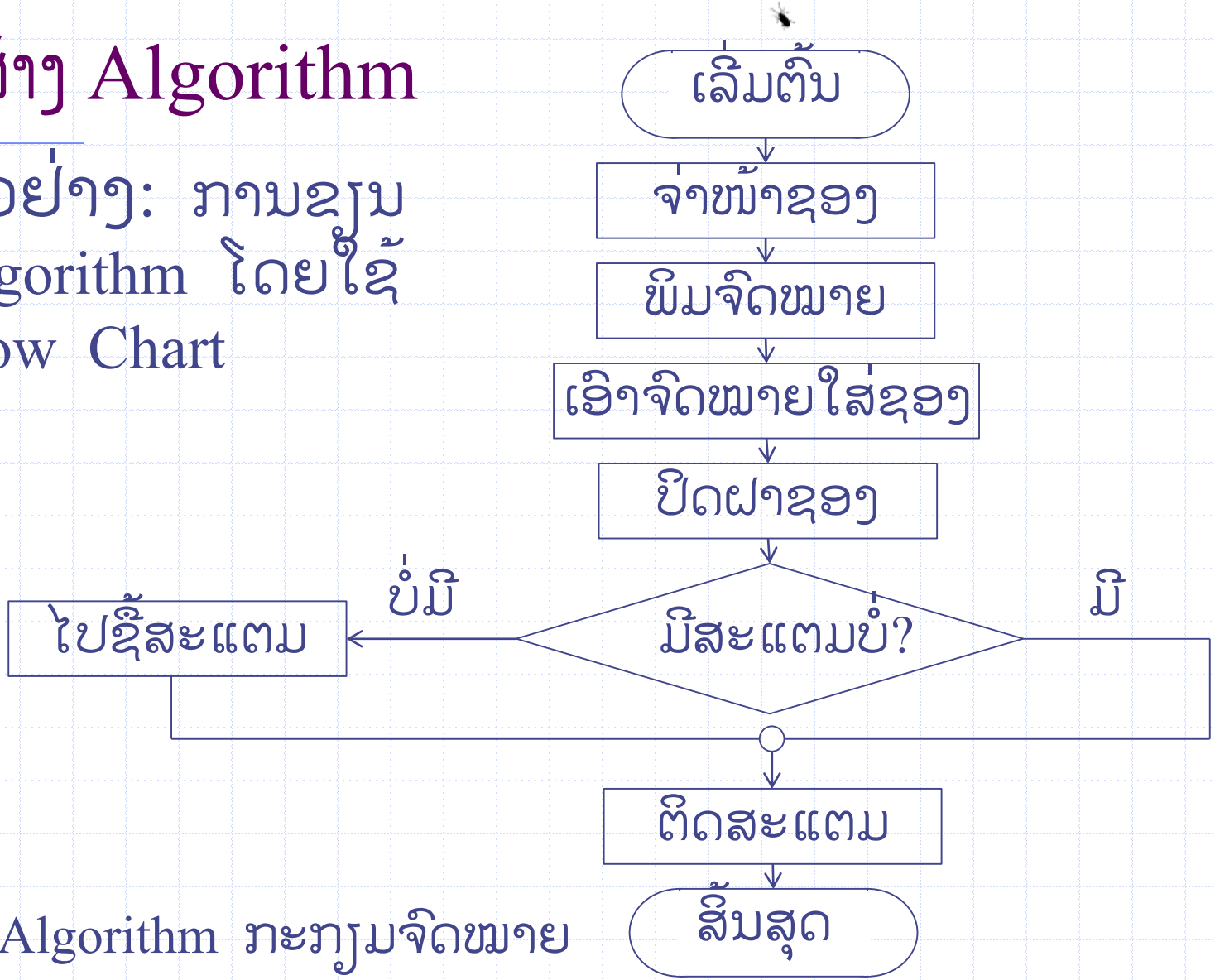
ວິທີສ້າງ Algorithm

- ຕົວຢ່າງລຳດັບຂັ້ນຕອນຂອງຄຳສັ່ງ
  - Algorithm ການລວມລາຄາສິນຄ້າໂດຍໃຊ້ຈັກຄິດໄລ່ເລກ
    1. ເປີດຈັກຄິດໄລ່ເລກ
    2. ເຮັດວຽກດັ່ງຕໍ່ໄປນີ້ຊ້ຳໆຕາມຈຳນວນສິນຄ້າ
       1. ພິມລາຄາສິນຄ້າ
       2. ກົດເຄື່ອງໝາຍ +
    3. ກົດເຄື່ອງໝາຍ =
    4. ບັນທຶກລາຄາທັງໝົດ
    5. ປິດເຄື່ອງຈັກຄິດໄລ່ເລກ

ການຂຽນ Algorithm ໂດຍໃຊ້ພາສາມະນຸດ

ວິທີສ້າງ Algorithm

- ຕົວຢ່າງ: ການຂຽນ Algorithm ໂດຍໃຊ້ Flow Chart



ການຂຽນ Algorithm ກະກຽມຈົດໝາຍ

ວິທີສ້າງ Algorithm

- Pseudo code ມີລັກສະນະຄ້າຍຄືພາສາອັງກິດ, ຢູ່ເຄິ່ງກາງລະຫວ່າງພາສາອັງກິດ ແລະ ພາສາຄອມພິວເຕີ
- ໃຊ້ໃນການອະທິບາຍລັກສະນະຂອງໂຄງສ້າງຂໍ້ມູນ ແລະ ການເຮັດວຽກຂອງລຳດັບຂັ້ນຕອນຂອງຄຳສັ່ງທີ່ໄດ້ສ້າງຂຶ້ນ

ລະຫັດ Pseudo

- ຕົວຢ່າງ: ການຂຽນ Algorithm ໂດຍໃຊ້ Pseudo Code

Algorithm Summation

1. SUM = 0
2. INPUT(value 1)
3. INPUT(value 2)
4. INPUT(value 3)
5. SUM = value 1+value 2 + value 3
6. OUTPUT(SUM)

   END

ການຂຽນ Algorithm ການບວກເລກ 3 ໂຕ

ພື້ນຖານທາງຄະນິດສາດ

- ພື້ນຖານທາງຄະນິດສາດທີ່ມັກຖືກໃຊ້ເປັນປະຈຳໃນການສຶກສາກ່ຽວກັບ Data Structure ແລະ Algorithm
  - Exponents ແລະ Exponential
  - Logarithms
  - Factorial
  - Modular
  - Fibonacci Number
  - Series

Abstract Data Type (ADT)

- ADT ແມ່ນການປະກາດຄຸນລັກສະນະຂອງໂຄງສ້າງຂໍ້ມູນ ແລະ ກຸ່ມຂອງການດຳເນີນການທັງໝົດ ທີ່ສາມາດກະທຳໄດ້ກັບໂຄງສ້າງຂໍ້ມູນນັ້ນ ໂດຍໃຊ້ຫຼັກການທາງຕັກກະ ແລະ ຄະນິດສາດ
- ໂດຍຫລັກການຂອງ ADT
  - ຜູ້ໃຊ້ບໍ່ຈຳເປັນຕ້ອງຮູ້ວ່າ ADT ນັ້ນມີວິທີໃນການດຳເນີນການໄດ້ຢ່າງໃດ
  - ຮູ້ພຽງແຕ່ວ່າ ADT ນັ້ນສາມາດເຮັດຫຍັງໄດ້ແດ່

Abstract Data Type (ADT)

- ຕົວຢ່າງ ADT ຂອງ Queue

AbstractDataType Queue

{

Instances:

Ordered list of elements; one end is called the Front; the other is the Rear

Operations

isEmpty() : Return true if the queue is empty, return false otherwise

getFrontElement() : Return the Front element of the queue

getRearElement() : Return the Rear element of the queue

put(n) : Add element n at the Rear of the queue

remove() : Remove an element from the front of the queue and return it

}

Abstract Data Type (ADT)

- ຕົວຢ່າງ ADT ຂອງ AlphaString

AbstractDataType AlphaString
{
Instances:
Value Definition; English alphabet: a-z or A-Z
Operations
makeStr() : return the value of AlphaString
isEqual() : Return true if two AlphaString is equal otherwise retuen false
length() : Return the length of the AlphaString
append() : return the value of AlphaString that is appended from substring
substr() : return the value of substring in the AlphaString
pos() : display the first position of substring in the AlphaString
}

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

- ຈຳນວນຖ້ວນ (Integer)
  - ຕາມມາດຕະຖານຂອງ ANSI ສຳຫຼັບຕົວເລກຖ້ວນຂະໜາດ 16 ບິດ

ຊະນິດ	ຊ່ວງຂອງຄ່າ	ຈຳນວນບິດ
Int	-32768 – 32767	16
Unsigned int	0 – 65535	16
Short int	-32768 – 32767	16
Long int	-2147483648 – 2147483647	32
Unsigned long int	0 – 4294967295	32

- ຈຳນວນຈິງ (Real)
  - ຈຳນວນຈິງຈະໃຊ້ຈຳນວນບິດທັງໝົດ 32 ບິດ ຊຶ່ງໃນນັ້ນບິດຊ້າຍມືສຸດ ແມ່ນບິດເຄື່ອງໝາຍ, ອີກ 8 ບິດຕໍ່ມາເປັນບິດກຳລັງ ແລະ 23 ບິດທີ່ເຫຼືອ ແມ່ນບິດຈຳນວນຫຼັງຈຸດ

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

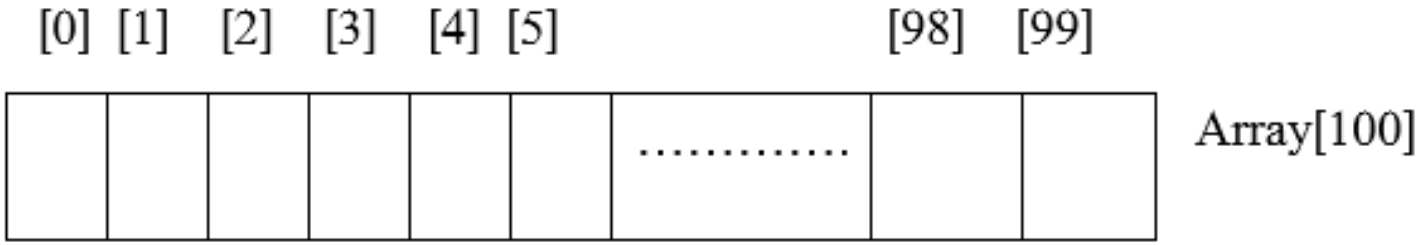
- ຕົວອັກສອນ (Character)
  - ຕາມມາດຕະຖານຂອງ ASCII, ອັກສອນແຕ່ລະຕົວຈະມີຂະໜາດ 8 ບິດ
- String
  - ເປັນຂໍ້ມູນປະເພດຂໍ້ຄວາມຍາວໆ
- Boolean
  - ເປັນປະເພດຂໍ້ມູນທີ່ມີຄ່າ ເປັນ TRUE = 1 ຫຼື FALSE = 0 ທີ່ມີຂະໜາດ 8 ບິດ

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

- Array
  - ເປັນໂຄງສ້າງຂໍ້ມູນທີ່ສາມາດເກັບຂໍ້ມູນປະເພດດຽວກັນ, ເປັນການເກັບຂໍ້ມູນທີ່ລຽງກັນເປັນແຖວທີ່ມີຂະໜາດຈຳກັດ
  - ມີໂຄງສ້າງເປັນເສັ້ນຊື່
  - ສາມາດເຂົ້າຫາ ຫຼື ອ້າງອີງຂໍ້ມູນໄດ້ໂດຍໃຊ້ index
  - ປະເພດຂອງ Array
    - Array 1 ມິຕິ
    - Array 2 ມິຕິ
    - Array 3 ມິຕິ

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

- ອາເຣ 1 ມິຕິ (One-Dimensional Array)
  - ເປັນໂຄງສ້າງອາເຣທີ່ງ່າຍທີ່ສຸດ ຊຶ່ງເກັບຂໍ້ມູນລຽນຕິດຕໍ່ກັນເປັນແຖວ



  - ຕຳແໜ່ງທຳອິດໃນອາເຣ ເອີ້ນວ່າ ຕຳແໜ່ງຖານ (Base Address) ຂອງ ອາເຣ

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

- ອາເຣ 2 ມິຕິ (Two-Dimensinal Array)
  - ມີລັກສະນະເປັນແບບມາຕິຣດ (Matrix) ຄືຕາຕະລາງ (Table) ເຊິ່ງປະກອບມີແຖວ (Row) ແລະ ຖັນ (Column)

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

- ອາເຣ 2 ມິຕິ (Two-Dimensinal Array)
  - ການອ້າງອີງເຖິງຂໍ້ມູນໃນອາເຣ 2 ມິຕິ ເຮັດໄດ້ໂດຍການກຳນົດລະຫວ່າງ ແຖວ ແລະ ຖັນທີ່ຂໍ້ມູນຢູ່ໂດຍໃຊ້ດັດຊະນີເຊັ່ນ: a[1][3] ໝາຍເຖິງ ຂໍ້ມູນ ໃນອາເຣທີ່ຢູ່ແຖວ 1 ຖັນ 3.
  - ການເອົາຂໍ້ມູນອອກຈາກອາເຣ 2 ມິຕິ ຫຼື ການເອົາຂໍ້ມູນເຂົ້າເກັບໃນອາເຣ ຈະຕ້ອງໃຊ້ດັດຊະນີ 2 ຕົວ ເພື່ອບອກທີ່ຢູ່ເຊັ່ນ:
    - x = a[i][j] ໝາຍເຖິງ ເອົາຄ່າຈາກອາເຣ a ແຖວທີ່ i ຖັນ j ໄປເກັບໃນຕົວປ່ຽນ x
    - a[i][j] = x ໝາຍເຖິງ ເອົາຄ່າຈາກຕົວປ່ຽນ x ເກັບໃນອາເຣ a ແຖວທີ່ i ຖັນທີ່ j

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

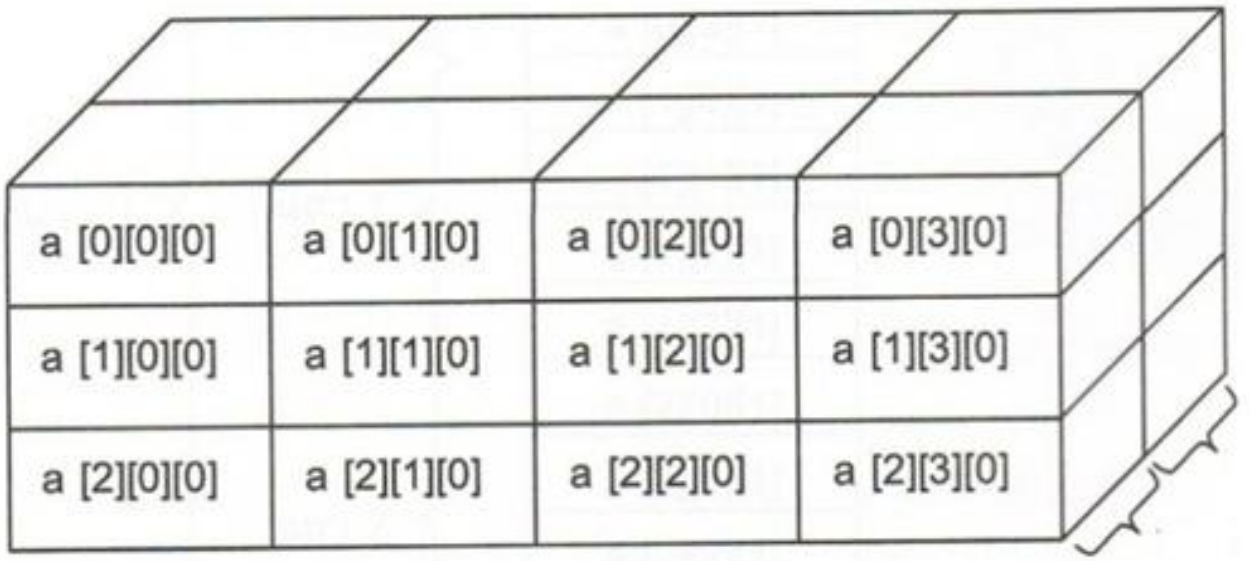
- ອາເຣ 2 ມິຕິ (Two-Dimensinal Array)
  - ວິທີ່ການອ້າງເຖິງຂໍ້ມູນແຕ່ລະຕົວໃນອາເຣເປັນດັ່ງນີ້

	ຖັນ 0	ຖັນ 1	ຖັນ 2	ຖັນ 3
ແຖວ 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
ແຖວ 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
ແຖວ 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

ອາເຣຂະໜາດ 3x4

ຊະນິດຂໍ້ມູນພື້ນຖານ ແລະ Array

- ອາເຣ 3 ມິຕິ (Three-Dimensinal Array)
  - ເປັນການເກັບຂໍ້ມູນທີ່ພົວພັນກັບ 3 ທິດທາງຄື ທາງແຖວ, ທາງຖັນ ແລະ ທາງສູງ



Recursive Function

- ເປັນ Function ທີ່ສາມາດເອີ້ນໃຊ້ຕົວເອງໄດ້ ໂດຍຮູບແບບຂອງມັນຈະຄືກັບ Function ທຳມະດາ
- ໂດຍສ່ວນໃຫຍ່ຈະຂຽນໂປຣແກຣມ Recursive ເພື່ອແກ້ໄຂບັນຫາທີ່ມີການກຳໜົດແບບເອີ້ນຊ້ຳ(recursive define) ເຊັ່ນການຊອກຫາຄ່າຂອງ n! , Fibonacci ແລະ ອື່ນໆ

Recursive Function

- n! = n * (n-1)! ໂດຍ n>0
- ມີຂໍ້ກຳໜົດວ່າ 0! =1

ຕົວຢ່າງ
ຈົ່ງຊອກຫາຄ່າຂອງ 3!
⇒ 3! = 3 * 2 * 1 * 0! = 6

ລອງຄິດ : ຈົ່ງຂຽນ Function ເພື່ອຊອກຫາຄ່າ Factorial ທີ່ຜູ້ໃຊ້ເປັນຜູ້ກຳໜົດ ໂດຍໃຊ້ For ຫຼື While ເຊັ່ນ ຊອກຫາຄ່າ 7!

Recursive Function

ຕົວຢ່າງ

ພິຈາລະນາ $3! = 3 * 2!$ $\quad n! = n * (n-1)!$

$2 * 1!$

$1 * 0!$

- ຈະເຫັນໄດ້ວ່າເມື່ອໄດ້ຄ່າ 3 ແລ້ວຍັງ 2! ແລະ ເມື່ອໄດ້ 2 ແລ້ວຍັງ 1! ເປັນເຊັ່ນນີ້ໄປເລື້ອຍໆ ການຄຳນວນໃນລັກສະນະແບບນີ້ຈະໃຊ້ Recursive Function ໃນການຄຳນວນ
- ສິ່ງທີ່ຄວນຈື່: ຜູ້ຂຽນຕ້ອງວິເຄາະໃຫ້ໄດ້ວ່າ Function ຈະເຊົ້າເອີ້ນຊ້ຳເມື່ອໃດ ໃນກໍລະນີນີ້ Function ຈະເຊົ້າເອີ້ນຊ້ຳເມື່ອ $n \leq 1$

Recursive Function

```
#include <stdio.h>
long factorial(int);
void main(){
        int i;
        printf("Enter Number : ");
        scanf("%d",&i);
        printf("\n %d! = %ld",i,factorial(i));
}
long factorial(int GetInput){
        if (GetInput<=1)
                return(1);
        else
                return(GetInput*factorial(GetInput-1));
}
```

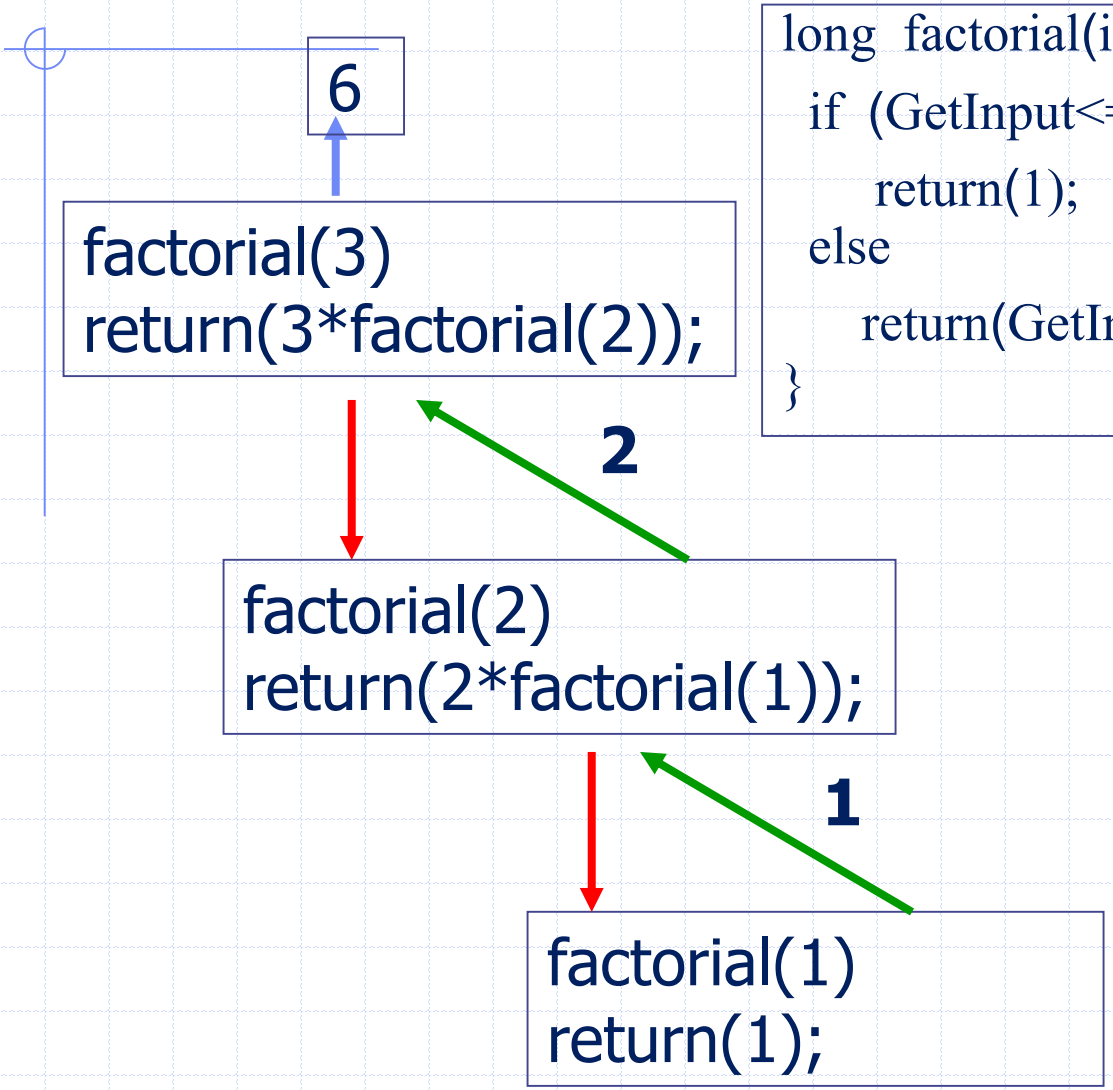
Recursive Function

```
long factorial(int GetInput){
 if (GetInput<=1)
  return(1);
 else  return(GetInput*factorial(GetInput-1));
}
```

ຫລັກການເຮັດວຽກຂອງ Function ຄືມັນຈະຊອກຫາຄ່າ factorial ໂດຍການເອີ້ນໃຊ້ Function factorial(3)

factorial(3) ສົ່ງຄ່າ (3 * factorial(3-1)) ກັບໄປຫາທີ່ຢູ່ທີ່ເອີ້ນມັນ
factorial(2) ສົ່ງຄ່າ (2 * factorial(2-1)) ກັບໄປຫາທີ່ຢູ່ທີ່ເອີ້ນມັນ
factorial(1) ສົ່ງຄ່າ 1 ກັບໄປຫາທີ່ຢູ່ທີ່ເອີ້ນມັນ
ດັ່ງນັ້ນ factorial(2) = 2*1 = 2, factorial(3) = 3*2 = 6

ອະທິບາຍ factorial(3) ດ້ວຍຮູບ




```
long  factorial(int  GetInput){
  if  (GetInput<=1)
       return(1);
  else
       return(GetInput*factorial(GetInput-1));
}
```