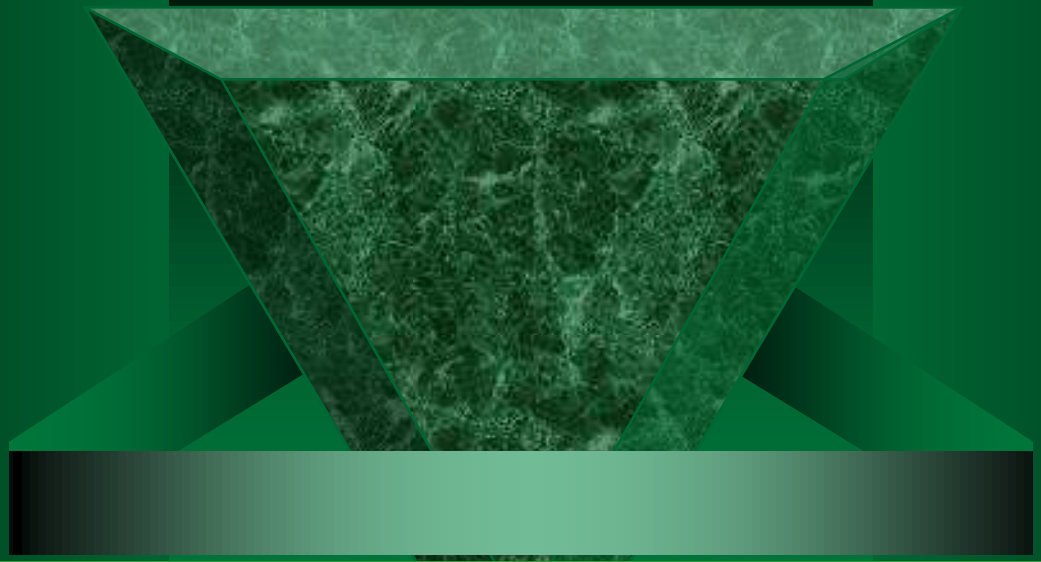
Apirak :



6.1. Function

The Objectives

- Build structured programs that are divided into function
- Describe the flow of execution in a program.
- Describe what the “scope of variable” means?
- Pass data to functions
- Get value from function using return

Let consider the app.

```
#include <iostream>
 using namespace std;
main()
{
 int choice;  // variable for user input
 int i;       // variable for loops and output
 do // loop until a valid choice is entered
  {
  cout << "Which series do you wish to display?\n";
  cout << "1 - Odd numbers from 1 to 30\n";
  cout << "2 - Even numbers from 1 to 30\n";
  cout << "3 - All numbers from 1 to 30\n";
  cin >> choice;  // get choice from user
  if ((choice < 1) || (choice > 3))
   {  // if invalid entry, give message
     cout << "Choice must be 1, 2, or 3\n";
   }
 } while ((choice < 1) || (choice > 3));
 switch (choice)
   {
    case 1:
      for (i = 1; i <= 30; i = i + 2)
       cout << i << ' ';
      cout << endl;
      break;
    case 2:
      for (i = 2; i <= 30; i = i + 2)
       cout << i << ' ';
      cout << endl;
      break;
    case 3:
      for (i = 1; i <= 30; i++)
        cout << i << ' ';
      cout << endl;
      break;
   }
  return 0;
}

```

Apirak :

ตัวอย่าง 1

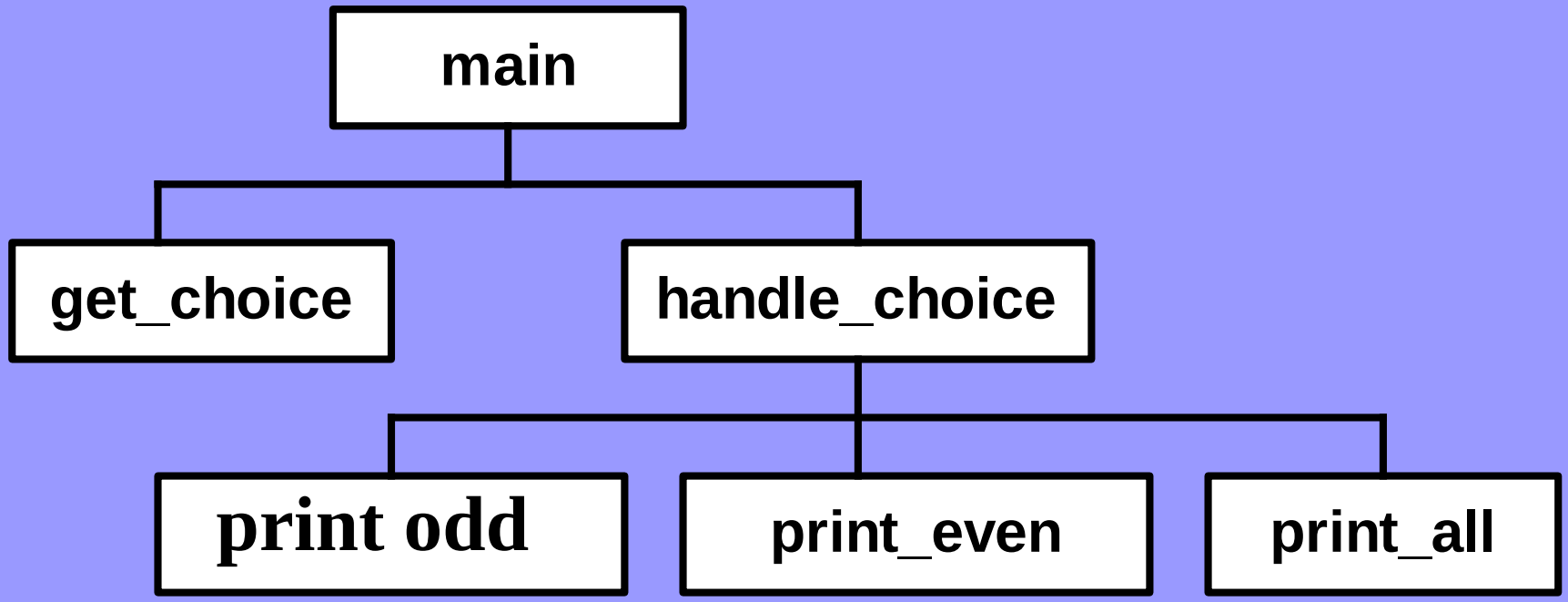
many activities contained in one program (complexity)

โปรแกรมรับข้อมูล 1(พิมพ์เลขคี่),2(พิมพ์เลขคู่,3(พิมพ์ทั้หมด)

```
C:\My Documents\sc107\files>series
Which series do you wish to display?
1 - Odd numbers from 1 to 30
2 - Even numbers from 1 to 30
3 - All numbers from 1 to 30
1
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29
Which series do you wish to display?
1 - Odd numbers from 1 to 30
2 - Even numbers from 1 to 30
3 - All numbers from 1 to 30
2
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
```

```
#include<iostream.h>  //example1_1.cpp
main()
{
 int choice;  // variable for user input
 int i;       // variable for loops and output
 do // loop until a valid choice is entered
  {
  cout << "Which series do you wish to display?\n"
  cout << "1 - Odd numbers from 1 to 30\n";
  cout << "2 - Even numbers from 1 to 30\n";
  cout << "3 - All numbers from 1 to 30\n";
  cin >> choice;  // get choice from user
  if ((choice < 1) || (choice > 3))
   {  // if invalid entry, give message
     cout << "Choice must be 1, 2, or 3\n";
   }
 } while ((choice < 1) || (choice > 3));
```

```
 switch (choice)
    {
     case 1:
       for (i = 1; i <= 30; i = i + 2)
         cout << i << ' ';
       cout << endl;
       break;
     case 2:
       for (i = 2; i <= 30; i = i + 2)
         cout << i << ' ';
       cout << endl;
       break;
     case 3:
       for (i = 1; i <= 30; i++)
          cout << i << ' ';
       cout << endl;
       break;
    }
  return 0;
}
```



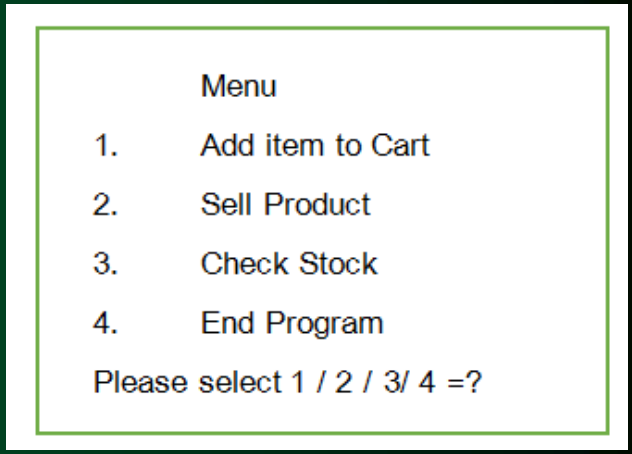
- *This example if we divide our program to sub-activities like above diagram. it make easier.*
- *C++ supports the idea of dividing program to sub-activities,called function.*

ตัวอย่าง 2

Duplicate activity

ให้นักศึกษาเขียนโปรแกรมขายสินค้าและพิมพ์ใบเสร็จรับเงิน
โดยข้อมูลในระบบมีไม่เกิน **100** รายการ
ข้อมูลขายสินค้าด้วย

- รหัสสินค้า(id) เช่น “AB001”
- ชื่อสินค้า(name) เช่น “CAFÉ”
- ราคาขาย(cost) เช่น 30.6
- จำนวนสินค้าในสต๊อก (instock) เช่น 5



EX1 จากตัวอย่างข้างต้นให้ นักศึกษาปรับโปรแกรม โดยให้มีการตรวจดังนี้

กรณี กด 1 : ให้ตรวจสอบว่ารหัสที่ป้อน มีซ้ำหรือไม่ ถ้าไม่มีจึงจะอนุญาตให้ผู้ใช้ป้อนข้อมูล

กรณี กด 2: ให้ตรวจสอบว่ารหัสสินค้าที่ผู้ใช้ป้อนมีอยู่ในระบบหรือไม่ กรณีไม่พบให้แสดงข้อความ “product not exist”

กรณี กด 3 ให้แสดงรายการทั้งหมด และแสดงยอดรวมของ cost

Check existing products activity

แบบฝึกหัด 2

จงเขียนโปรแกรมต้องการนำระเบียนลูกค้า(customer) ของสหกรณ์ออมทรัพย์รามคำแหง เก็บในตัวแปรแถวลำดับ 1 มิติ ระเบียนลูกค้าประกอบด้วย รหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก (Deposit)

Menu

1. Open a new Account
2. Deposit
3. Withdraw
4. Compound interest
5. Quit

Please select 1 / 2 / 3 / 4 / 5 =?

เลือก 1 เป็นการเปิดบัญชีใหม่

ระบบจะแสดงหน้าจอให้ผู้ใช้ ป้อนรหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก (Deposit) โดยรหัสบัญชีต้องไม่ซ้ำกัน

เลือก 2 เป็นการฝากเงิน

โดยผู้ใช้ป้อนรหัสบัญชี(id)ที่ต้องการ ระบบจะตรวจสอบเพื่อค้นหาระเบียนลูกค้า เพื่อให้ผู้ใช้งานป้อนยอดฝาก เพื่อปรับปรุงบัญชีให้ถูกต้อง

เลือก 3 เป็นการถอนเงิน

โดยผู้ใช้ป้อนรหัสบัญชี(id)ที่ต้องการ ระบบจะตรวจสอบเพื่อค้นหาระเบียนลูกค้า เพื่อให้ผู้ใช้งานป้อนยอดถอนแต่ต้องมีเงินติดบัญชีอย่างน้อย 100 บาท ระบบทำการปรับปรุงบัญชีให้ถูกต้อง

เลือก 4 เป็นการคิดดอกเบี้ยทบต้น

โดยระบบจะให้ผู้ใช้ป้อนอัตราดอกเบี้ยต่อปี(rate) ทางแป้นพิมพ์ เพื่อคิดดอกเบี้ยทบต้นของทุกบัญชี ซึ่งกระทำทุกวัน ได้ดอกเบี้ยเป็นรายวัน โปรแกรมจะแสดงเงินฝากเก่า ดอกเบี้ยที่ได้รับ และเงินฝากใหม่ปรากฏบนจอภาพ

เลือก 5 เป็นการจบการทำงานซ้ำ

```cpp
using namespace std;

struct Account {
    char id[10];
    char name[100];
    char addr[100];
    float deposit;
};

int main()
{
    struct Account cust[500];
    int n = 0;
    int choice;
    char id[10];
    bool bfound=false;
    float deposit;
    int i;
    do
    {
        cout << "\n1.Open Acc";
        cout << "\n2.Deposit Acc";
        cout << "\n3.Withdraw Acc";
        cout << "\n4.Compute Interest";
        cout << "\n5.exit\n";

        cin >> choice;

    switch(choice)
    {
        case 1:
                cout << "\n do Open Acc";
                cout << "\n id Acc =? ";
                cin >> id;
                bfound=false;
                for(int i=0 ; i< n ; i++)
                {
                    if( strcmp(id, cust[i].id)==0)
                    {
                        cout << "id Acc found =? " << id;
                        bfound = true;
                        break;
                    }
                }
                if(bfound==false)
                {
                    strcpy( cust[n].id,id );
                    cout << "name Acc =? ";
                    cin >> cust[n].name;
                    cout << "addr Acc =? ";
                    cin >> cust[n].addr;
                    cout << "deposit Acc =? ";
                    cin >> cust[n].deposit;
                    n++;
                }

                break;
```

Apirak :

```
60              case 2:
61                      {
62                          cout << "\n do Deposit  Acc";
63                          cout << "\n id Acc =? ";
64                          cin >> id;
65                          for( i=0 ; i< n ; i++)
66                          {
67                              if( strcmp(id, cust[i].id)==0)
68                              {
69                                  cout << "before deposit Acc =? " << cust[i].deposit ;
70                                  cout << "\nnew deposit Acc =? ";
71                                  cin >> deposit;
72                                  cust[i].deposit += deposit;
73                                  cout << "\nafter deposit Acc =? " << cust[i].deposit;
74                                  break;
75                              }
76                          }
77                          if(i==n) {
78                              cout << "id Acc not found =? " << id;
79                          }
80                      }
81                      break;
82              case 3:
83                      cout << "\n do Withdraw Acc";
84                      break;
85              case 4:
86                      cout << "\n do Interest Acc";
87                      break;
88          } //switch(choice)
89
90      }  while(choice!=5);
91
92      return 0;
93  }
```

Apirak :

Rewrite code :Case 2

```
60          case 2:
61              {
62                  cout << "\n do Deposit  Acc";
63                  cout << "\n id Acc =? ";
64                  cin >> id;
65
66                  bfound=false;
67                  for(int i=0 ; i< n ; i++)
68                  {
69                      if( strcmp(id, cust[i].id)==0)
70                      {
71                          cout << "id Acc found =? " << id;
72                          bfound = true;
73                          break;
74                      }
75                  }
76                  if(bfound==false)
77                  {
78                      cout << "id Acc not found =? " << id;
79                  }
80                  else
81                  {
82                      cout << "before deposit Acc =? " << cust[i].deposit ;
83                      cout << "\nnew deposit Acc =? ";
84                      cin >> deposit;
85                      cust[i].deposit += deposit;
86                      cout << "\nafter deposit Acc =? " << cust[i].deposit;
87
88                  }
89              }
90              break;
```

EX1 จากตัวอย่างข้างต้นให้ นักศึกษาปรับโปรแกรม โดยให้มีการตรวจดังนี้

กรณี กด 1 : ให้ตรวจสอบว่ารหัสที่ป้อน มีซ้ำหรือไม่ ถ้าไม่มีจึงจะอนุญาตให้ผู้ใช้ป้อนข้อมูล

กรณี กด 2: ให้ตรวจสอบว่าสินค้าที่ผู้ใช้ป้อนมีอยู่ในระบบหรือไม่ กรณีไม่พบให้แสดงข้อความ “product not exist”

กรณี กด 3 ให้แสดงรายการทั้งหมด และแสดงยอดรวมของ cost

```
product List:
Id          name    cost  stock
--------------------
AB001  CAFÉ   30.6    3
```

Guidelines for building a program with function

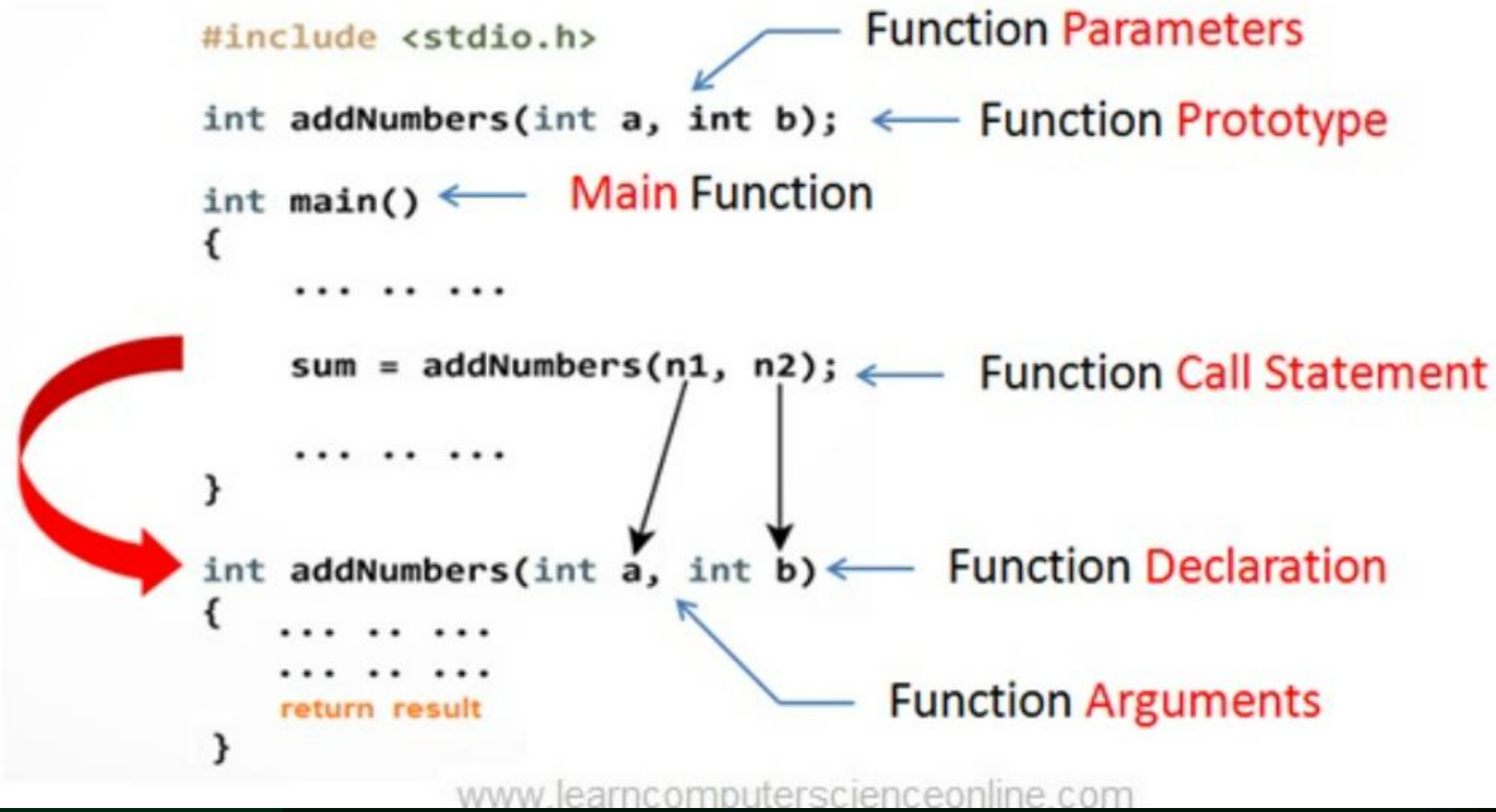
- ***Organization**: A large program is easier to read and modify if it is logically organized into function. Once a single function is tested and performs properly,you can set it aside and concentrate on problem areas.*

- ***Autonomy**: the function does not depend on data or code outside the function any more than necessary.*

Type of Functions(routines) in Programming lang.

- *Library function*
  - *standard library (build-in): coming with programming lang. such as setw(-) , printf(-) in c/c++*
  - *Third-party library : coding by other programmer teams such buying , opensource.*
- *Customize function : coding by yourself such as main(-) in c/c++*

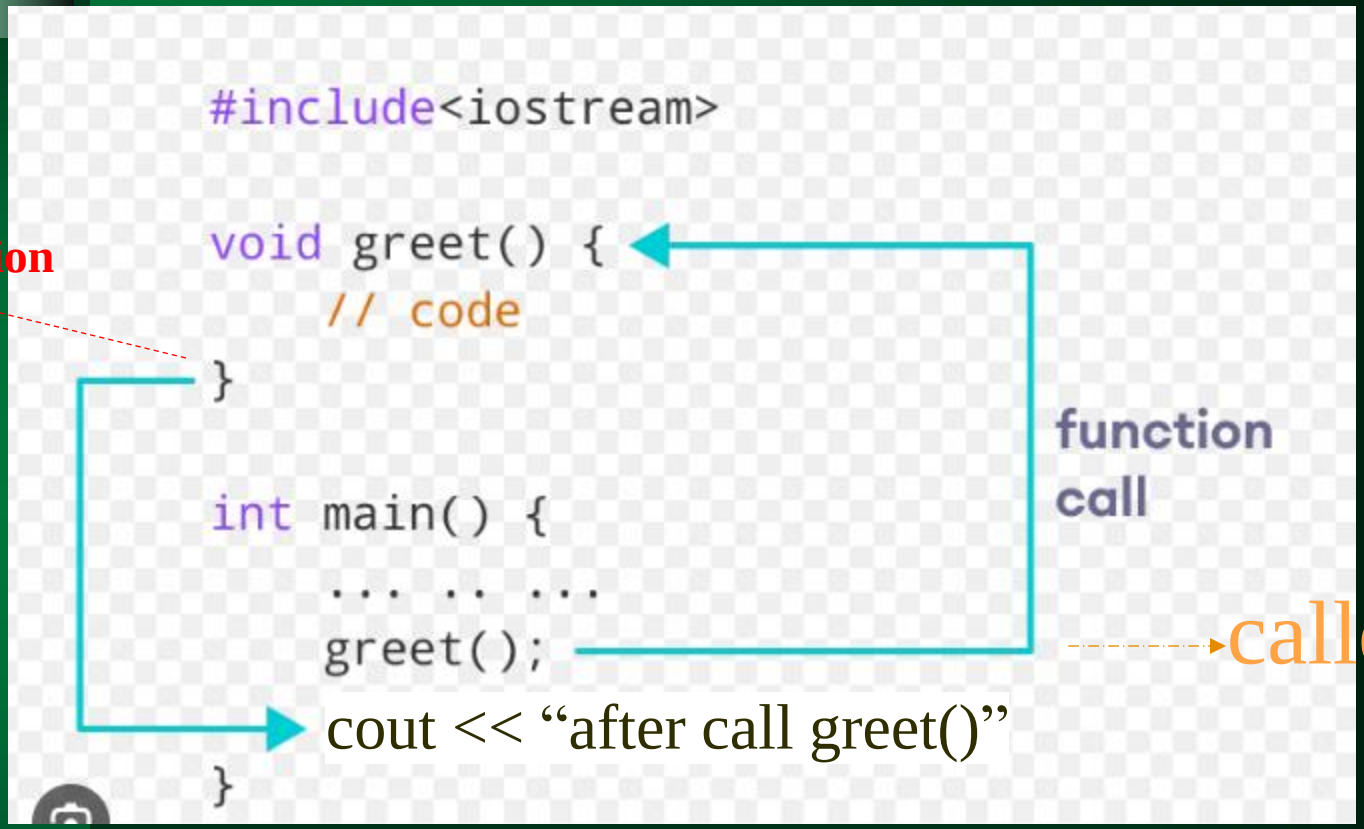
program organization on c/c++ :



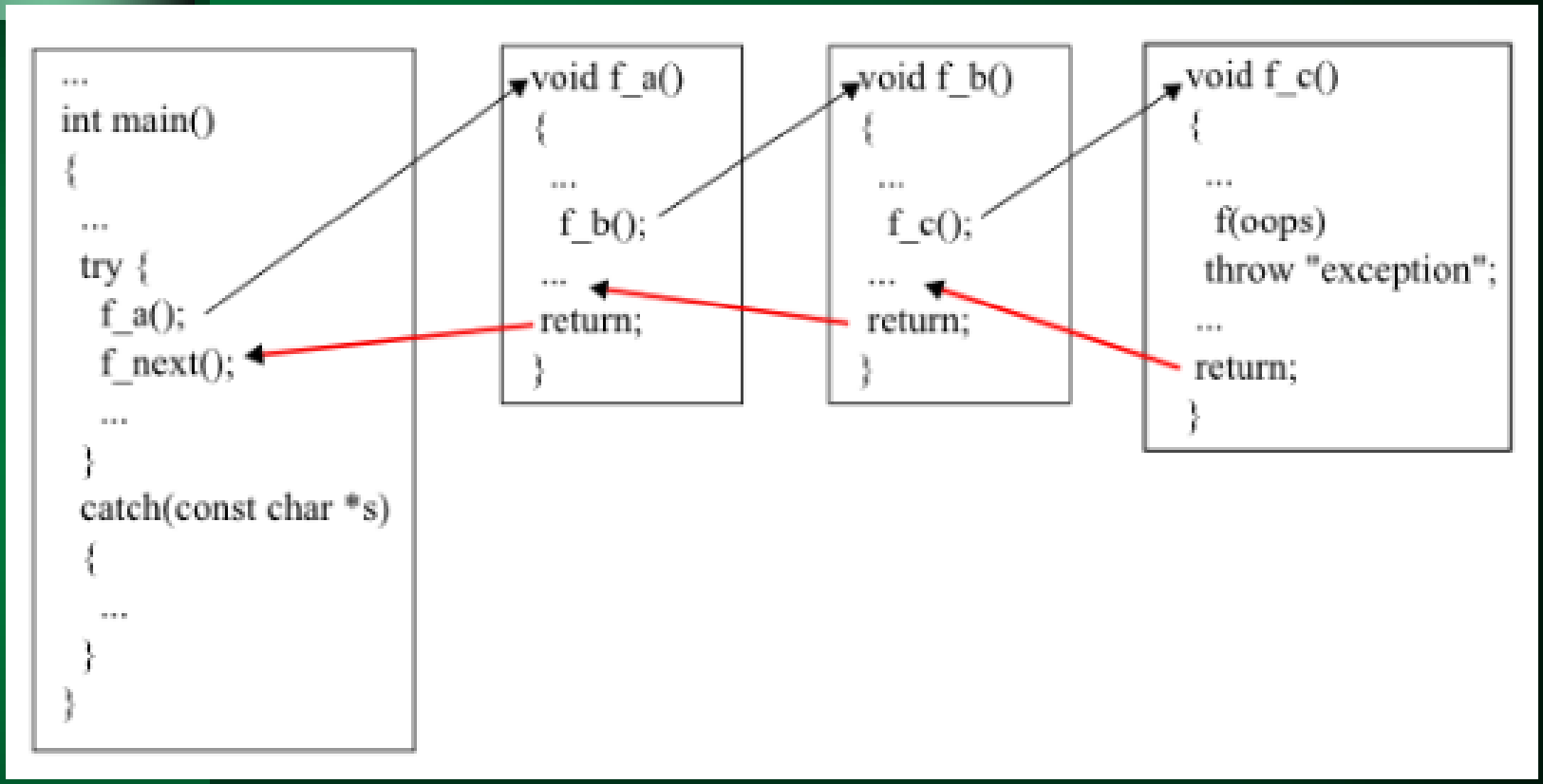
https://www.learncomputerscienceonline.com/procedural-programming/

Execution flows (1)

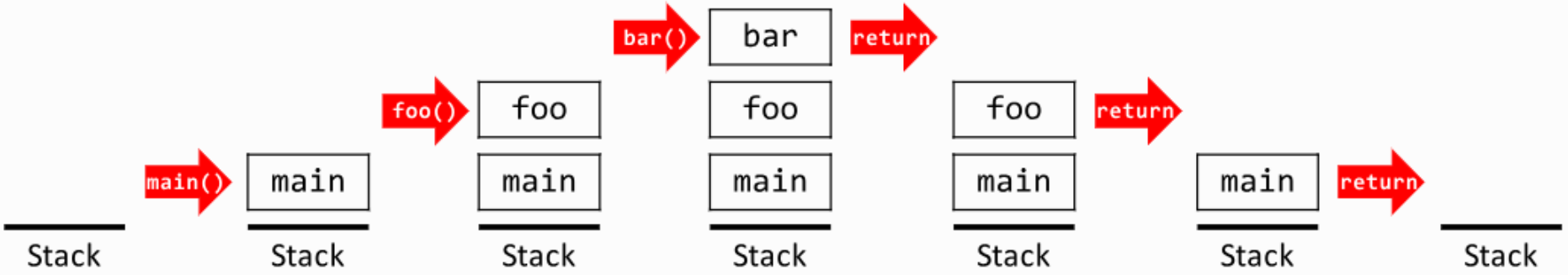
End fucntion



Execution flows (2)



Execution flows (3)



Operating System → call

```
void bar() {
}

void foo() {
  bar();
}

int main() {
  foo();
}
```

https://eecs280staff.github.io/notes/02_ProceduralAbstraction_Testing.html

Execution flows : exiting of function in "C/C++ with return statement

terminating function with "}"

```
#include<iostream>

void greet() {
    // code
}

int main() {
    ... .. ...
    greet();
    ... .. ...
}
```

function call

terminating function with "return" instruction

```
#include<stdio.h>

int sum(int a, int b)
{
    return a+b;
     a = a-b;
}
int main()
{
    int ans;
    ans = sum(10,5);
    printf("10 + 5 = %d",ans);
    return 0;
}
```

Called function

Calling function

Execution will start from here after return

Characteristic of function

1. No return value function (void function)
   (procedure subroutine/ void function)
2. Return value function
   (funtion subroutine)

Function Syntax in c/c++

```
return_type function_name ( parameter 1, parameter 2, . . . . )
    {

        //function body

    }
```

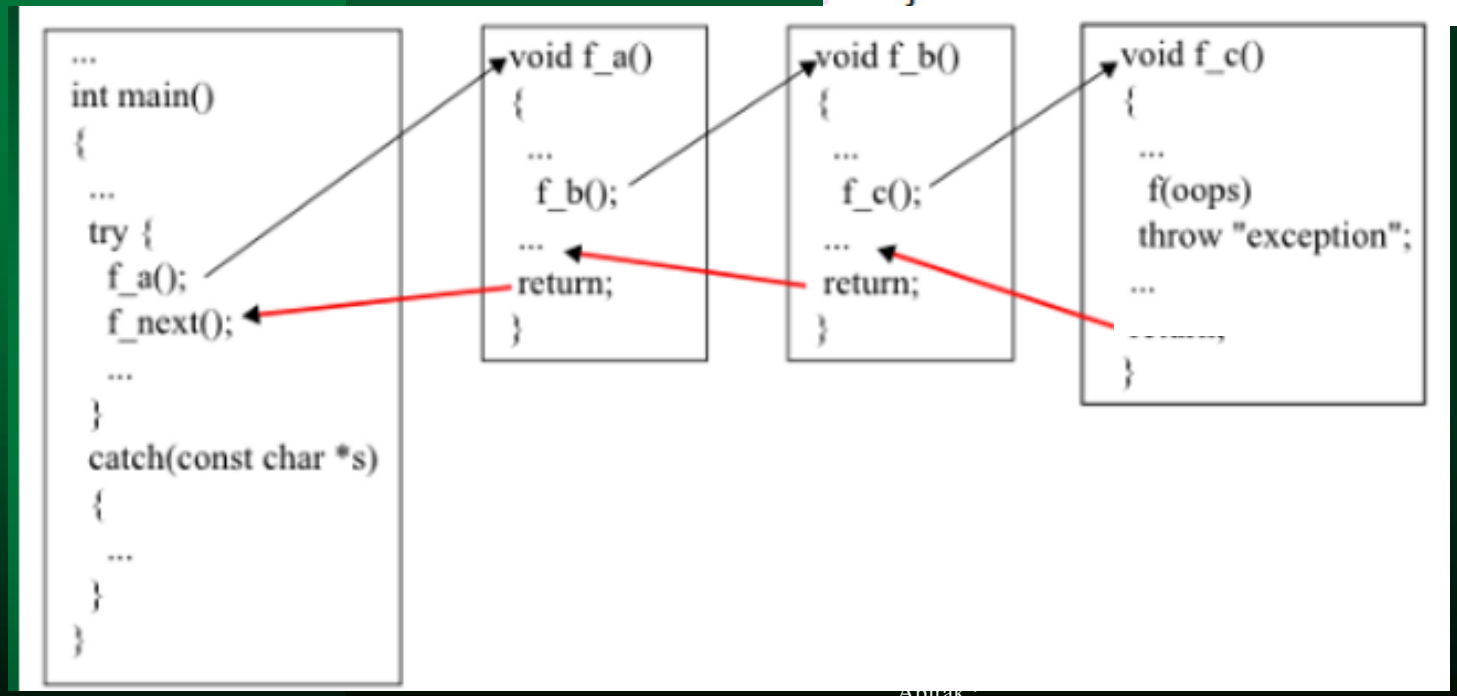
```
// Function to print program title to screen.
void print_title (  )
 {
  cout << "Tennis Tournament Scheduler Program\n";
  cout << "By Jennifer Baker\n";
 }
```

```
12  // Function to print program title to screen.
13  int get_value_ten (  )
14   {
15       return 10;
16   }
```

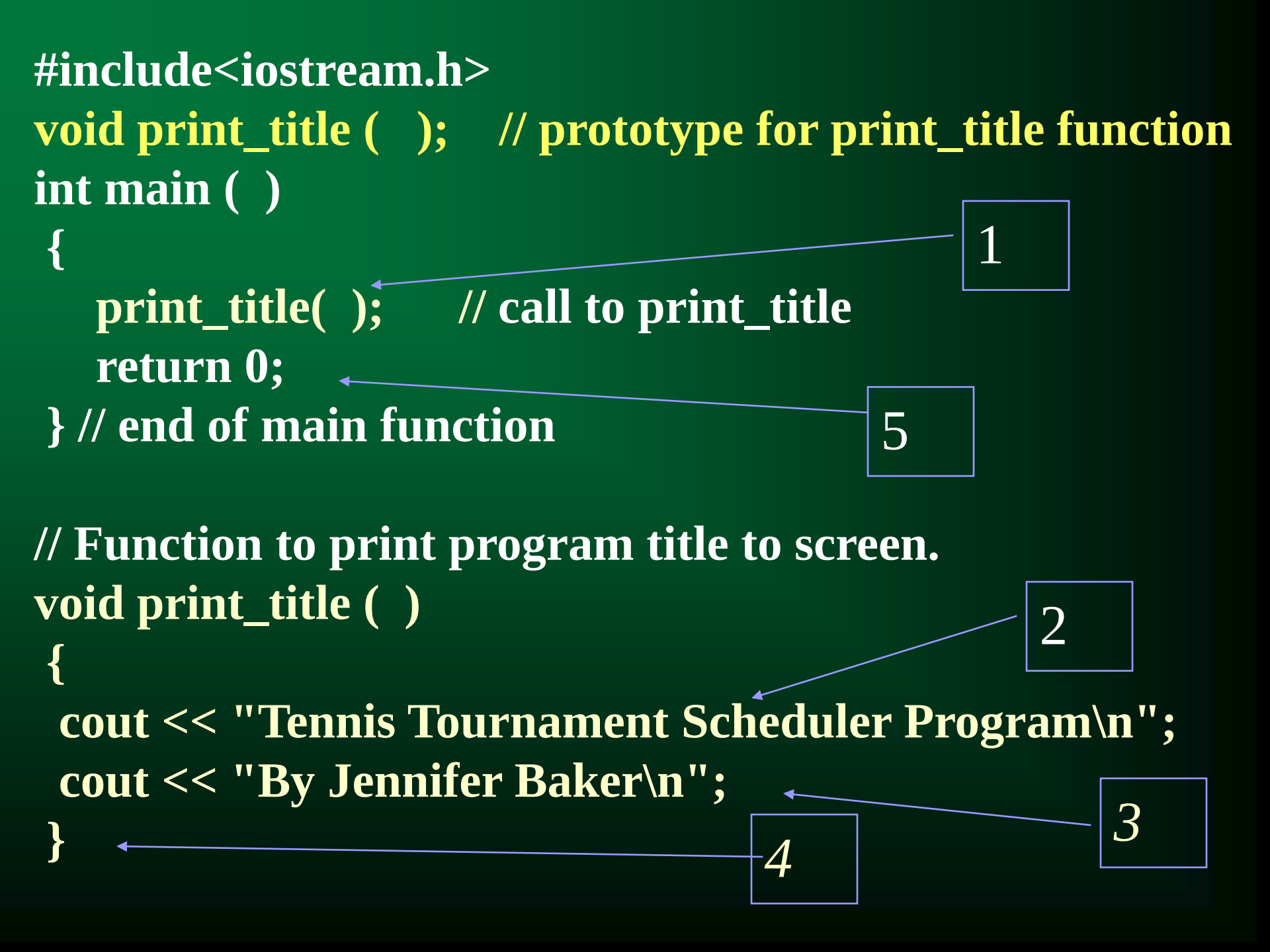
c/c++ no return value function (void function)

Variables declare in function body

```
void fname (formal-parameter-declaration-list)
{
    local variable declarations
    executable statements
    return ;
}
```



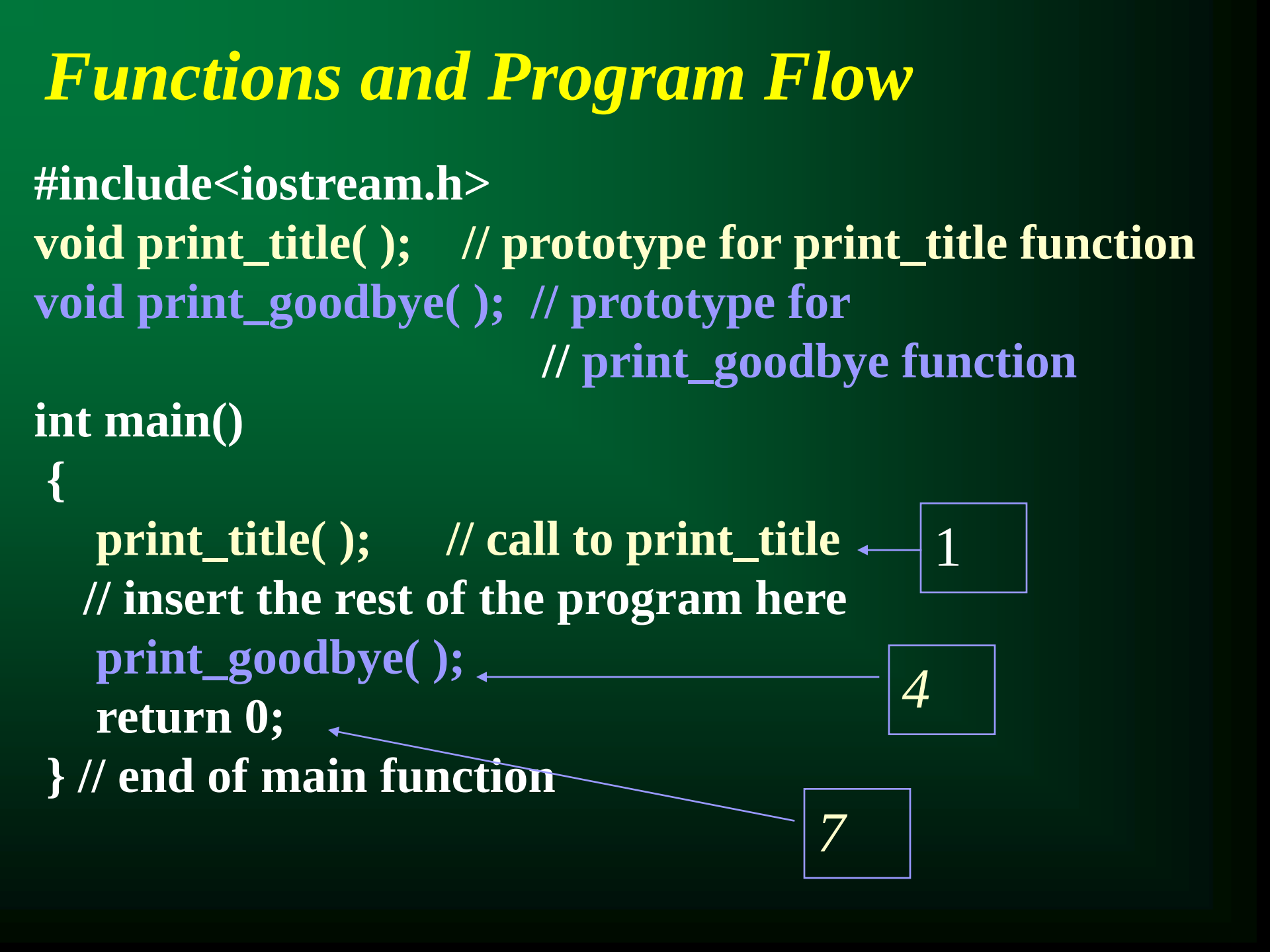
```
#include<iostream.h>
void print_title (   );    // prototype for print_title function
int main (  )
 {
    print_title(  );      // call to print_title
    return 0;
 } // end of main function

// Function to print program title to screen.
void print_title (  )
 {
  cout << "Tennis Tournament Scheduler Program\n";
  cout << "By Jennifer Baker\n";
 }
```



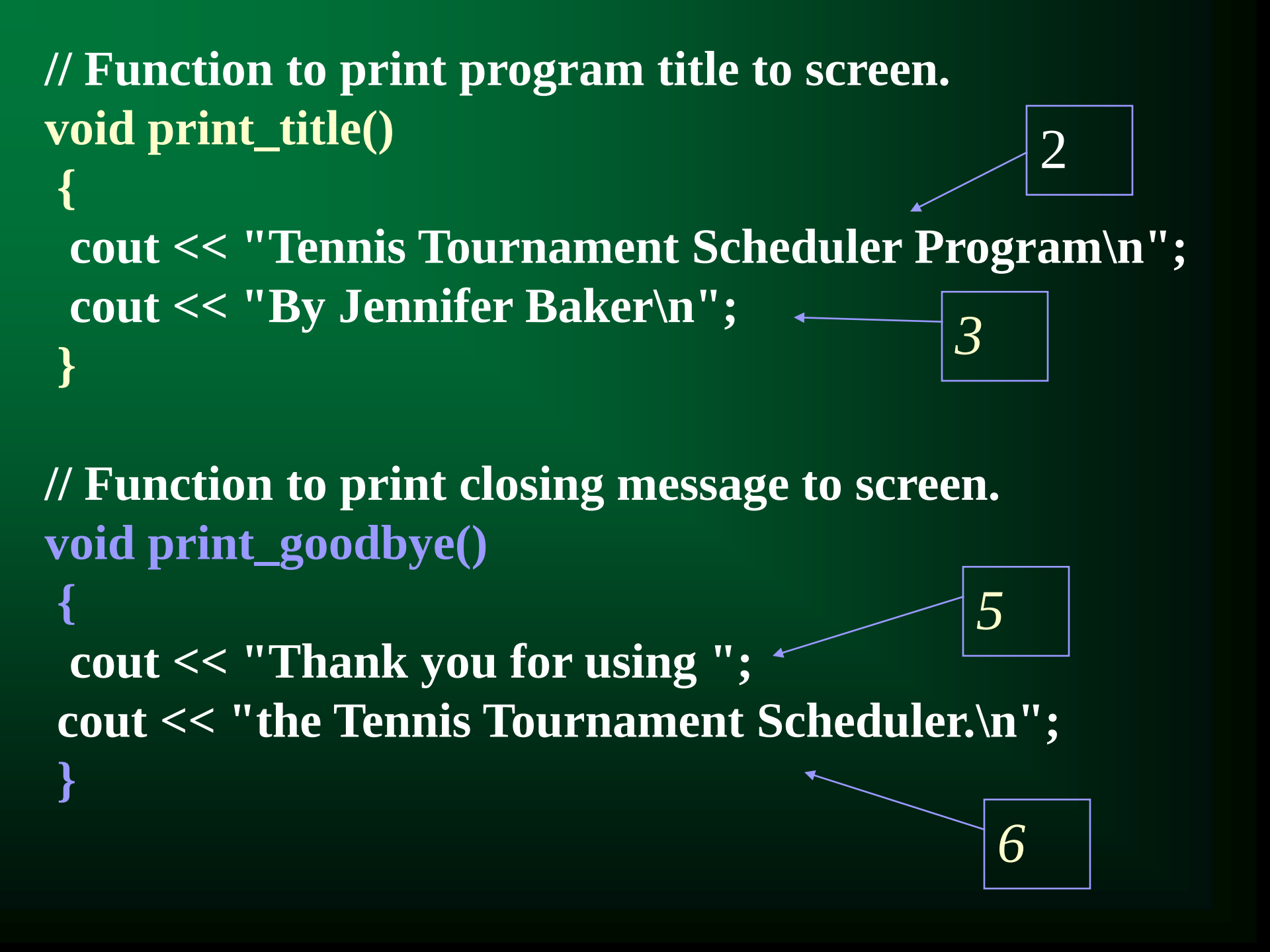
```
C:\My Documents\c\Release>test
Tennis Tournament Scheduler Program
By Jennifer Baker
```

Functions and Program Flow

```
#include<iostream.h>
void print_title( );    // prototype for print_title function
void print_goodbye( );  // prototype for
                                  // print_goodbye function
int main()
 {
    print_title( );      // call to print_title
   // insert the rest of the program here
    print_goodbye( );
    return 0;
 } // end of main function
```



```
// Function to print program title to screen.
void print_title()
 {
  cout << "Tennis Tournament Scheduler Program\n";
  cout << "By Jennifer Baker\n";
 }

// Function to print closing message to screen.
void print_goodbye()
 {
  cout << "Thank you for using ";
 cout << "the Tennis Tournament Scheduler.\n";
 }
```



```
Tennis Tournament Scheduler Program
By Jennifer Baker
Thank you for using the Tennis Tournament Scheduler.
```

c/c++ return value function grammar

```
ftype  fname (formal-parameter-declaration-list)
{
        local variable declarations
        executable statements
        return value;
}
```

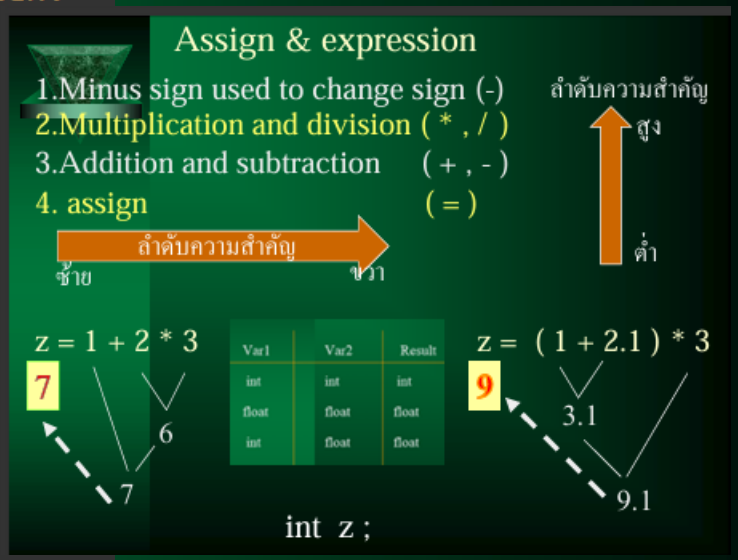
```
1  #include<iostream>
2  using namespace std;
3  int get_value_ten (   );    // prototype for pr
4  int main (  )
5   {
6       int x;
7
8       x = get_value_ten(  );      // call to pr
9       cout << x;
10      get_value_ten(  );      // call to print_
11      return 0;
12  } // end of main function
13
14 // Function to print program title to screen.
15 int get_value_ten (  )
16  {
17      return 10;
18  }
19
```

1. Ftype : int,float,char, Structure
2. Ftype : array (mention later).

- Using with “return value” statements

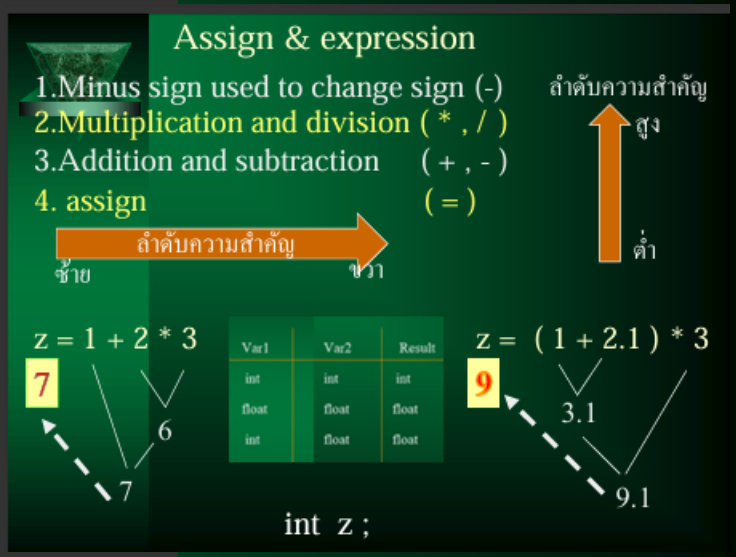
nain.cpp

```cpp
#include<iostream>
using namespace std;

// Function to print program title to screen.
int get_value_ten (  )
 {
     return 10;
 }

 int get_value_two()
 {
     return 2;
 }

int main (  )
 {
     int z;

     z = get_value_ten(  ) + get_value_two(  );
     cout << z << endl;


     return 0;
 } // end of main function

```



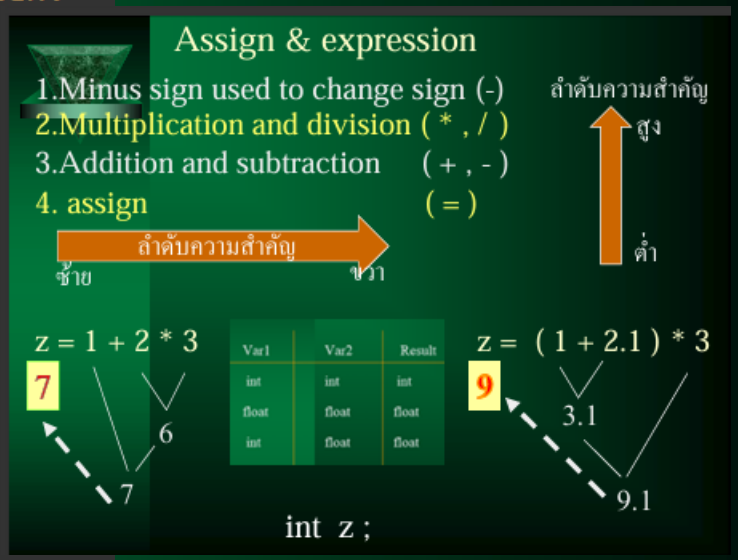
main.cpp

```
5  int get_value_ten (  )
6  {
7      return 10;
8      cout << "get_value_ten (  )" << 10;
9  }
10
11 int get_value_two()
12 {
13     return 2;
14 }
15
16 int main (  )
17 {
18     int z;
19
20     z = get_value_ten(  ) + get_value_two(  );
21     cout << z << endl;
22
23
24     return 0;
25 } // end of main function
26
27
28
```

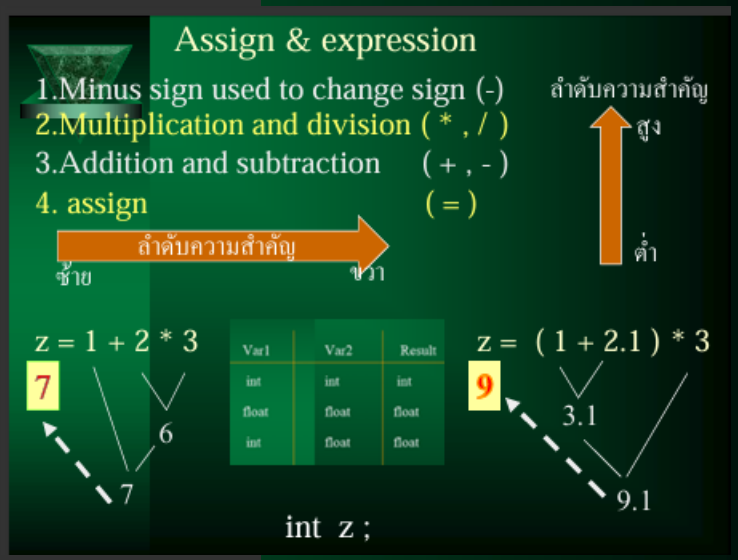


nain.cpp

```cpp
#include<iostream>
using namespace std;

// Function to print program title to screen.
int get_value_ten (  )
 {
     return 10;
 }

 int get_value_two()
 {
     return 2;
 }

int main (  )
 {
     int z;

     z = get_value_ten(  ) + get_value_two(  );
     cout << z << endl;


     return 0;
 } // end of main function

```



```cpp
#include<iostream>
using namespace std;

// Function to print program title to screen.
int get_value_ten (  )
 {
     return 10;
 }

 int get_value_two()
 {
     return 2;
 }

int main (  )
 {
     int z;

     z = get_value_ten(  ) + get_value_two(  )*100;
     cout << z << endl;


     return 0;
 } // end of main function

```



Inside "main" function

- *The C++ program has at least one function as main().*

```
int main( )
{
   // body of  program
   return 0 ;
}
```

Return type of function

Return value of function

- *when the program reaches the return 0; statement. the value zero is returned to the OS.*

Old complier versions allow "void" on main function

```
void main( )
{
   // body of  program


   return ;
}
```

Support "void" on main function

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World";
    return 0;
}
```

```
Hello World

...Program finished with exit code 0
Press ENTER to exit console.
```

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World";
    //return 0;
}
```

```
Hello World

...Program finished with exit code 0
Press ENTER to exit console.
```

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World";
    return 1;
}
```

```
Hello World

...Program finished with exit code 1
Press ENTER to exit console.
```

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World";
    return 2;
}
```

```
Hello World

...Program finished with exit code 2
Press ENTER to exit console.
```

Apirak :

- The value to be return is specified using the *return* statement and the function could return only one value.

```
char getmenuchoice( ) ;                    //prototype
char getmenuchoice( )
{
      char select;
      cout << " Menu" << endl;
     cout << "1. Add item to Cart " << endl;
     cout << "2. Sell product " << endl;
     cout << "3. Check stock " << endl;
     cout << "4. End program" << endl;
     cout << "Please select 1 / 2 / 3/ 4 =? ";
     cin >> select;
     return select;
}
int main ( ) {
   char selectMenu ;
   selectMenu = getmenuchoice( );
}
```

```
struct product {
  char id[10];
  char name[30];
  float cost;
  int instock;
};

struct product  getNewProduct() {
   struct product p;
   cout << "New product: " << endl;
   cout << "id =? ";  cin >> p.id;
   cout << "name=? "; cin >> p.name;
   cout << "cost=? "; cin >> p.cost;
   cout << "instock=? "; cin >> p.instock;
   return p;
}
int main() {
    struct product newProduct = getNewProduct();

}
```

return structure

Python allows a function return multiple values

main.py

```python
def operate(a, b):
    sum = a + b
    diff = a - b
    mul = a * b
    div = a / b
    return sum, diff, mul, div

n1 = 5
n2 = 10
sum, diff, mul, div = operate(n1, n2)
print(
    f"The sum is {sum}\n"
    f"The difference is {diff}\n"
    f"The multiplication gives {mul}\n"
    f"The division gives {div}\n"
)
```

Shell

```
The sum is 15
The difference is -5
The multiplication gives 50
The division gives 0.5
```

https://www.programiz.com/python-programming/online-compiler/

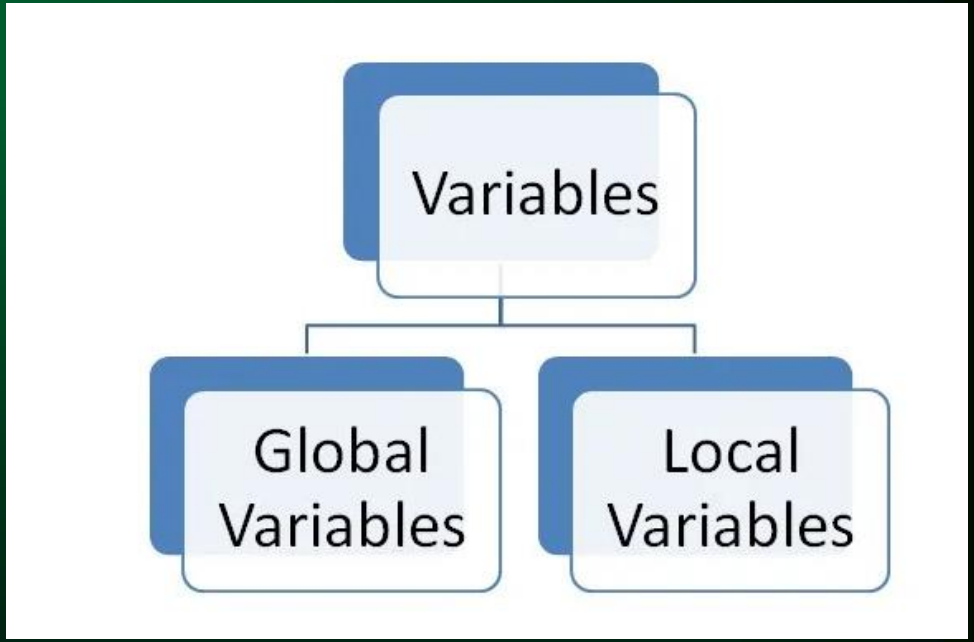
Scope of variable (Global & Local variable)

Scope determines the accessibility (visibility) of variables. c/c++ supports the following scope :

Local Block scope

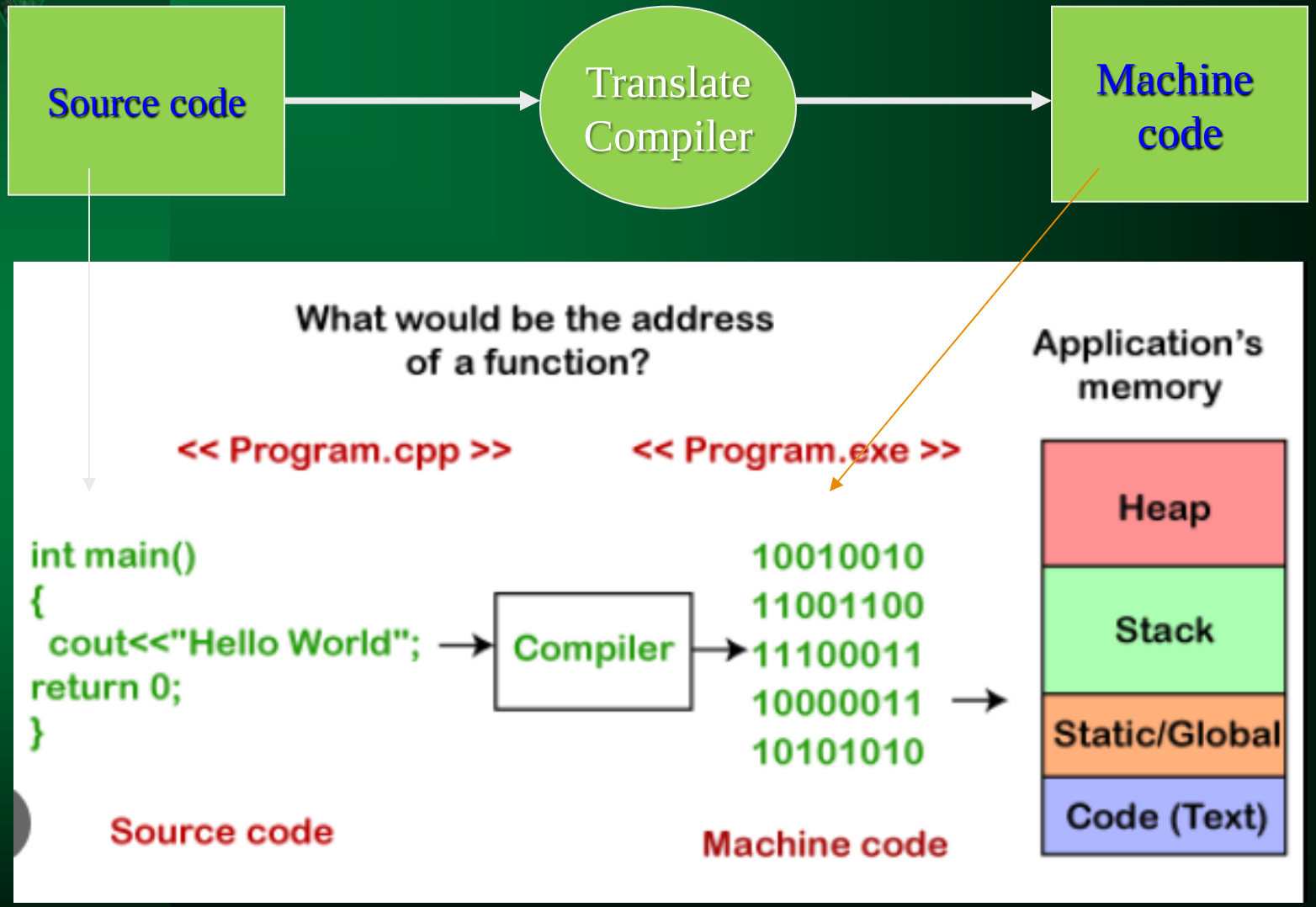
Local Function scope

Global Scope

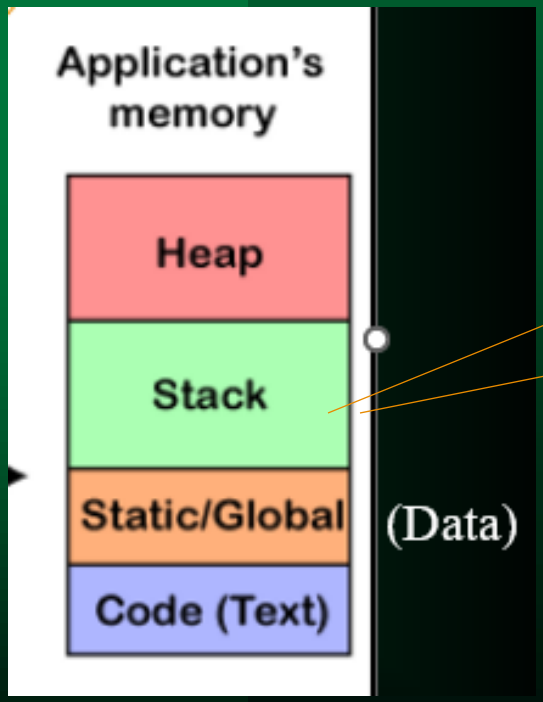


Recap:

Compiler is a tool that translate a source code to machine code



Global & Local Block & Local scope Property

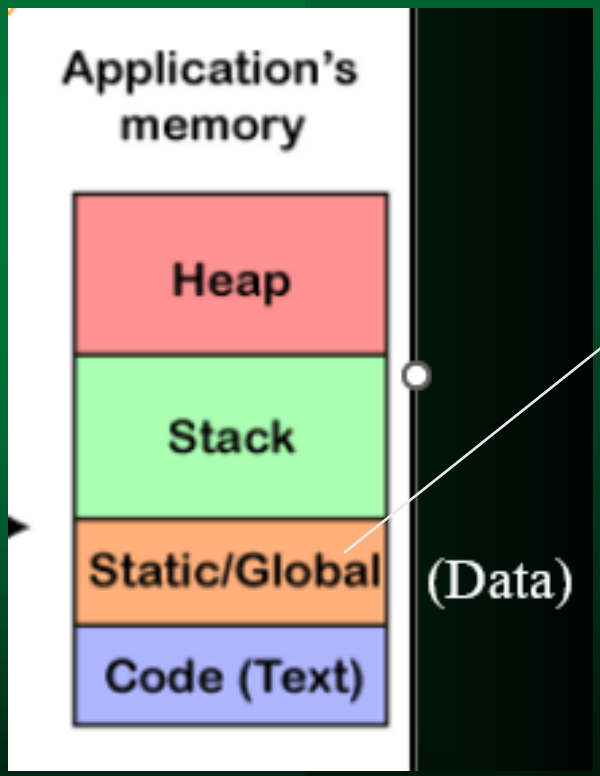
- Memories of local variables declared within a function are allocated on stack when the function or the block is starting to execute every times.
- The allocated memories (on stack) are free , when the function or block is existed(terminate)
- 

main.cpp

```
1  #include <iostream>
2  using namespace std;
3  #include <stdio.h>
4  int g = 100; //global variable
5  int main()
6  {
7      int my_num = 7;  //local function variable
8      {
9          int new_num = 10;   //local block variable
10         int my_num = 17;
11         printf("1. new_num is %d\n",new_num);
12         printf("2.my_num is %d\n",my_num);
13     }
14     printf("3.my_num is %d\n",my_num);
15     {
16         int new_num;
17         my_num = 7*2;
18         printf("5. new_num is %d\n",new_num);
19         printf("6.my_num is %d\n",my_num);
20     }
```

Global & Local Block & Local scope Property

- Memory of global variables are allocated on data(global) segment when the program starts.
- The memories are freed when the program terminates.



main.cpp

```
1  #include <iostream>
2  using namespace std;
3  #include <stdio.h>
4  int g = 100; //global variable
5  int main()
6  {
7      int my_num = 7;  //local function variable
8      {
9          int new_num = 10;   //local block variable
10         int my_num = 17;
11         printf("1. new_num is %d\n",new_num);
12         printf("2.my_num is %d\n",my_num);
13     }
14     printf("3.my_num is %d\n",my_num);
15     {
16         int new_num;
17         my_num = 7*2;
18         printf("5. new_num is %d\n",new_num);
19         printf("6.my_num is %d\n",my_num);
20     }
```

Block scope vs local scope

main.cpp

```cpp
using namespace std;
#include <stdio.h>
int main()
{
    int my_num = 7;
    {
        int new_num = 10;
        int my_num = 17;
        printf("1. new_num is %d\n",new_num);
        printf("2.my_num is %d\n",my_num);
    }
    printf("3.my_num is %d\n",my_num);
    {
        int new_num;
        my_num = 7*2;
        printf("5. new_num is %d\n",new_num);
        printf("6.my_num is %d\n",my_num);
    }
    printf("7.my_num is %d\n",my_num);
    return 0;
}
```

```
1. new_num is 10
2.my_num is 17
3.my_num is 7
5. new_num is 0
6.my_num is 14
7.my_num is 14
```

Block scope vs local scope

main.cpp

```
2  using namespace std;
3  #include <stdio.h>
4  int main()
5  {
6      int my_num = 7;
7      {
8          int new_num = 10;
9          int my_num = 17;
10         printf("1. new_num is %d\n",new_num);
11         printf("2.my_num is %d\n",my_num);
12     }
13     printf("3.my_num is %d\n",my_num);
14     {
15       //  int new_num;
16         my_num = 7*2;
17         printf("5. new_num is %d\n",new_num);
18         printf("6.my_num is %d\n",my_num);
19     }
20     printf("7.my_num is %d\n",my_num);
21     return 0;
22 }
23
```

input

Compilation failed due to following error(s).

```
main.cpp: In function 'int main()':
main.cpp:17:37: error: 'new_num' was not declared in this scope
   17 |         printf("5. new_num is %d\n",new_num);
      |                                     ^~~~~~~
```

Block scope vs local scope

```c
#include <iostream>
using namespace std;
#include <stdio.h>
int main()
{
    int my_num = 7;
    {
        int new_num = 10;
        int my_num = 17;
        printf("1. new_num is %d\n",new_num);
        printf("2.my_num is %d\n",my_num);
    }
    printf("3.my_num is %d\n",my_num);
    {
        int new_num;
        my_num = 7*2;
        printf("5. new_num is %d\n",new_num);
        printf("6.my_num is %d\n",my_num);
    }
    printf("7.my_num is %d\n",my_num);
    return 0;
}
```

```
1. new_num is 10
2.my_num is 17
3.my_num is 7
5. new_num is 0
6.my_num is 14
7.my_num is 14
```

main.cpp

```c
#include <stdio.h>
int main()
{
    int my_num = 7;
    {
        int new_num = 10;
        int my_num = 17;
        printf("1. new_num is %d\n",new_num);
        printf("2.my_num is %d\n",my_num);
    }
    printf("3.my_num is %d\n",my_num);
    {
        //int new_num = 10;
        my_num = 7*2;
        printf("5. new_num is %d\n",new_num);
        printf("6.my_num is %d\n",my_num);
    }
    printf("7.my_num is %d\n",my_num);
    return 0;
}
```

input

Compilation failed due to following error(s).

```
main.cpp: In function 'int main()':
main.cpp:17:37: error: 'new_num' was not declared in this scope
   17 |         printf("5. new_num is %d\n",new_num);
      |                                     ^~~~~~~
```

Apirak :

Global vs local scope

main.cpp

```
#include <iostream>
using namespace std;
#include <stdio.h>
int g =0 ;
void f1() {
    int f1_num;
    g++;
    printf("1. f1_g is %d\n",g);
    printf("2. f1_num is %d\n",f1_num);
    f1_num++;
}

void f2() {
    g += 10;
    printf("3. f2_g is %d\n",g);
}

void f3() {
    int g = 100;
    printf("4. f3_g is %d\n",g);
}
int main()
{
    printf("5. main_g is %d\n",g);
    f1();
    printf("6. main_g is %d\n",g);
    f2();
    printf("7. main_g is %d\n",g);
    f3();
    printf("8. main_g is %d\n",g);
    f1();
    printf("9. main_g is %d\n",g);
 }
```

```
5. main_g is 0
1. f1_g is 1
2. f1_num is 0
6. main_g is 1
3. f2_g is 11
7. main_g is 11
4. f3_g is 100
8. main_g is 11
1. f1_g is 12
2. f1_num is 100
9. main_g is 12
```

Global vs local scope

main.cpp

```cpp
#include <iostream>
using namespace std;
#include <stdio.h>
int g =0 ; //global
void f1() {
    int f1_num;
    g++;
    printf("1. f1_g is %d\n",g);
    printf("2. f1_num is %d\n",f1_
    f1_num++;
}

void f2() {
    f1_num++;   //local from f1
    g += 10;
    printf("3. f2_g is %d\n",g);
}

void f3() {
    int g = 100;
    printf("4. f3_g is %d\n",g);
}
int main()
{
    printf("5. main_g is %d\n",g);
    f1();
    printf("6. main_g is %d\n",g);
    f2();
    printf("7. main_g is %d\n",g);
    f3();
    printf("8. main_g is %d\n",g);
    f1();
    printf("9. main_g is %d\n",g);
 }

```

input

Compilation failed due to following error(s).

```
main.cpp: In function 'void f2()':
main.cpp:14:5: error: 'f1_num' was not declared in this scope
   14 |     f1_num++;   //local from f1
      |     ^~~~~~
```

```
4  using namespace std;
5  int  g = 10 , i = 3;    // global variable
6  void myfunction();
7  int main()
8   {
9      int j , k;
10     j = 2;
11     k = i + j;
12     cout <<"\n1.main k :" << k ;
13     cout <<"\n1.main i :" <<  i  ;
14     myfunction(  ); // call to myfunction
15     cout <<"\n2.main k :" << k ;
16     cout <<"\n2.main j :" << j ;
17     cout <<"\n2.main i :" <<  i  ;
18     g++;
19     myfunction(  ); // call to myfunction
20     cout <<"\n3.main k :" << k ;
21     cout <<"\n3.main j :" << j ;
22     cout <<"\n3.main g :" << g    ;
23     cout <<"\n4.main i :" <<  i  ;
24     return 0;
25  }
26 void myfunction()
27  {
28   int k;      // local variable
29   int g = 40;
30   k = i++; g = k ;
31   cout <<"\nmyfunction k :" << k ;
32   cout <<"\nmyfunction g :" << g    ;
33 }
34
```

k = ?

j = ?

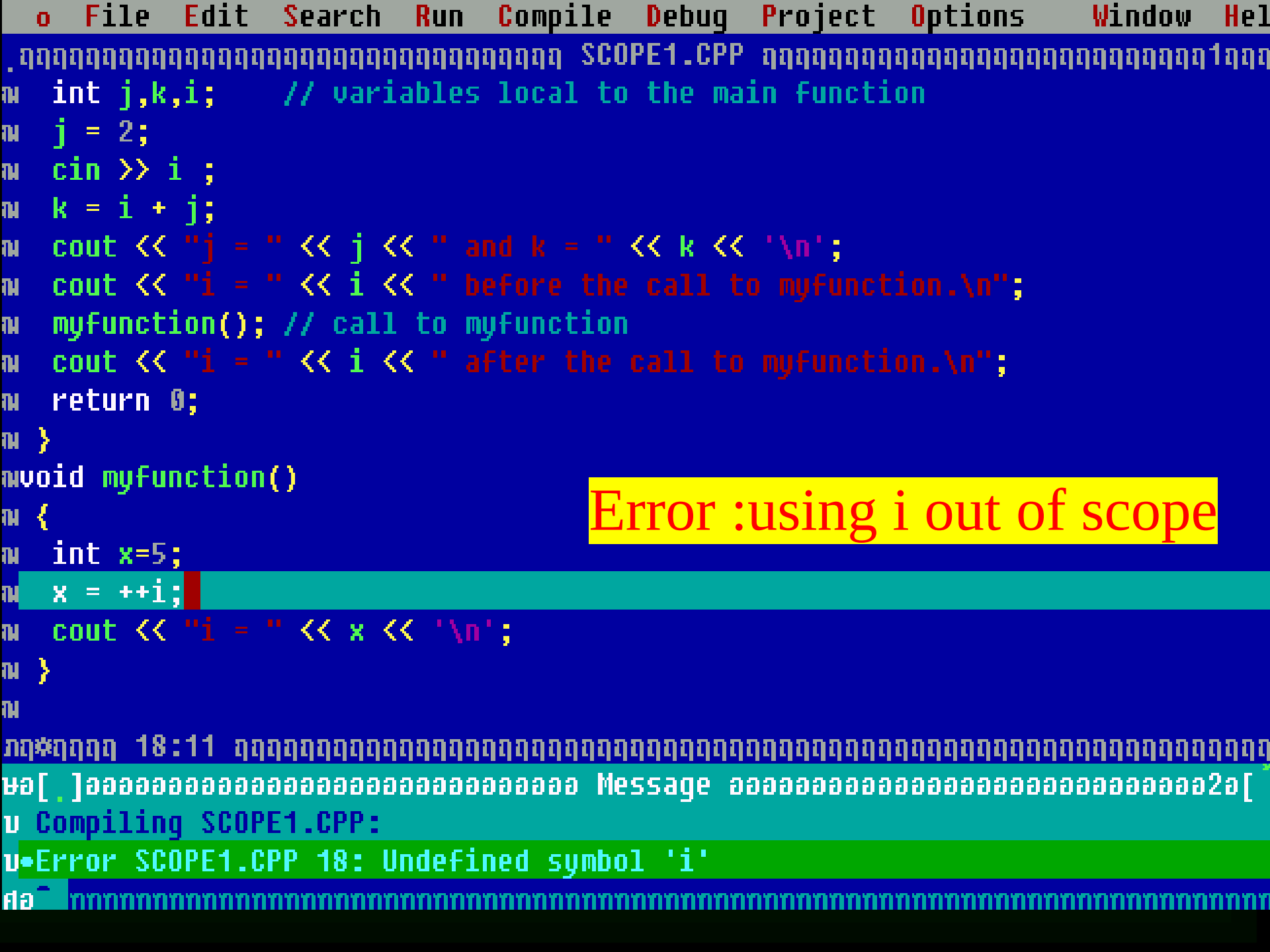
g = ?

i = ?

Apirak :

```
#include<iostream.h>
void myfunction();
int main()
 {
  int j,k,i;    // variables local to the main function
  j = 2;
  cin >> i ;
  k = i + j;
  cout << "j = " << j << " and k = " << k << '\n';
  cout << "i = " << i << " before the call to myfunction.\n";
  myfunction( ); // call to myfunction
  cout << "i = " << i << " after the call to myfunction.\n";
  return 0;
 }
void myfunction()
 {
  int x=5;
  x = ++i;
  cout << "i = " << x << '\n';
 }
```

i = ?


o File Edit Search Run Compile Debug Project Options Window Hel
SCOPE1.CPP
int j,k,i; // variables local to the main function
j = 2;
cin >> i ;
k = i + j;
cout << "j = " << j << " and k = " << k << '\n';
cout << "i = " << i << " before the call to myfunction.\n";
myfunction(); // call to myfunction
cout << "i = " << i << " after the call to myfunction.\n";
return 0;
}
void myfunction()
{
int x=5;
x = ++i;
cout << "i = " << x << '\n';
}
18:11
Message
Compiling SCOPE1.CPP:
Error SCOPE1.CPP 18: Undefined symbol 'i'
Error :using i out of scope

```
#include<iostream.h>
int  g = 10 , i = 3;    // global variable
void myfunction();
int main()
 {
   int j , k;    // variables local to the main function
   j = 2;     k = i + j;
  cout << "before g = " << g <<" \n";
  cout << "j = " << j << " and k = " << k << '\n';
  cout << "i = " << i
       << " before the call to   myfunction.\n";
  myfunction(  ); // call to myfunction
  cout << "i = " << i
      << " after the call to myfunction.\n";
  cout << "after g = " << g ;
  return 0;
 }
```

```
void myfunction()
 {
   int l;      // local variable
   int g = 40;
   l = ++i;
   cout << "l = " << l << '\n';
  cout << "The variable l is lost
        as soon as  myfunction exits.\n";

 cout << "g (myfunction) = "
        << g <<" \n";
 g = 80;
 }
```

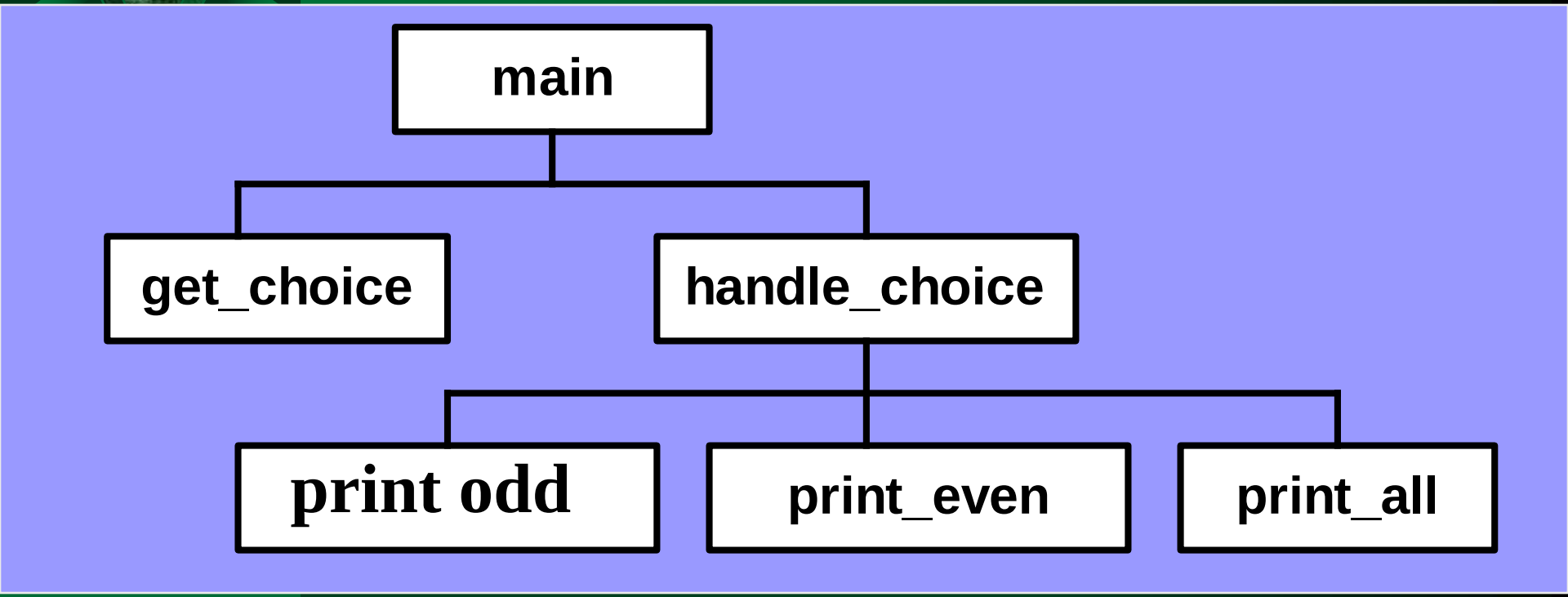
ผลของการ execute โปรแกรม คือ ?

ผลของการ execute โปรแกรม คือ ?

```
#include<iostream.h>
int  i = 3;    // global variable
void myfunction();
int main()
 {
    int j , k;    // variables local to the main function
    j = 2;
    k = i + j;
  cout << "j = " << j << " and k = " << k << '\n';
  cout << "i = " << i << " before the call to myfunction.\n";
  myfunction(  ); // call to myfunction
  cout << "i = " << i << " after the call to myfunction.\n";
  return 0;
 }
```

```
void myfunction()
{
    int l;      // local variable
    l = ++i;    // the variable i is accessible because i is global
                // the variable i is changed globally
    cout << "l = " << l << '\n';
    cout << "The variable l
                is lost as soon as myfunction exits.\n";
}
```

Home work



- rewrite a **`example1_1.cpp`** program by dividing activities to correspond with about diagram.

```cpp
#include <iostream>
using namespace std;

int getchoice() {
 int choice;  // variable for user input
 do // loop until a valid choice is entered
  {
     cout << "Which series do you wish to display?\n";
     cout << "1 - Odd numbers from 1 to 30\n";
     cout << "2 - Even numbers from 1 to 30\n";
     cout << "3 - All numbers from 1 to 30\n";
     cin >> choice;  // get choice from user
     if ((choice < 1) || (choice > 3))
      {  // if invalid entry, give message
         cout << "Choice must be 1, 2, or 3\n";
      }
  } while ((choice < 1) || (choice > 3));
  return choice;
}

void printOdd() {
   for (int i = 1; i <= 30; i = i + 2)
        cout << i << ' ';
   cout << endl;
}

void printEvent() {
   for (int i = 2; i <= 30; i = i + 2)
        cout << i << ' ';
   cout << endl;
}
```

```cpp
void printAll() {
   for (int i = 2; i <= 30; i = i + 2)
        cout << i << ' ';
   cout << endl;
}

int main()
{

    int choice = getchoice();
    switch (choice)
    {
     case 1:
        printOdd();
       break;
     case 2:
        printEvent() ;
       break;
     case 3:
        printAll( );
       break;
    }
  return 0;
}
```

Apirak :

7.2 parameter passing

Passing/Getting Data to and from function

- There are two ways to pass data to function in C++
  - Passing by value : send data to process on a function
  - Passing by reference : retrieve/update data form a function

7.2 Passing Data

- Memory of parameters on function are allocated on stack.
- They are allocated during a function is executing ,and are deallocated after the function is ended like local parameters.
- The initial values of parameters is from argument variables.

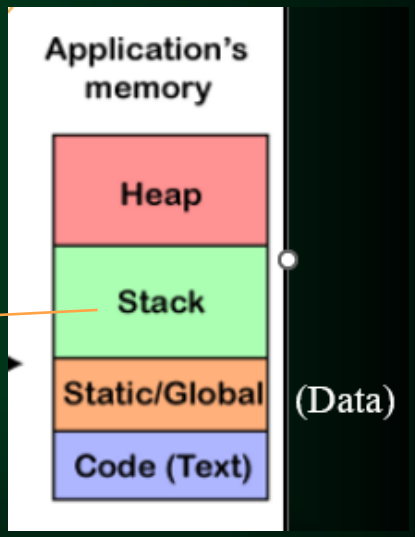
```
void pass_val ( int x , double &y ) ;
int main()
{ double y = 18.0 ;
  int x = 3 ;
  pass_val ( x , y )
  cout << x << y << endl ;
}
void pass_val ( int l , double &y )
{
     l = l*2;  y = y+2;
    cout << l << y << endl ;
}
```

arguments

-x = passed by values
-y = pass by reference

parameters

Apirak :



Grammar : Function Body

```
2  #include <iostream>
3
4  using namespace std;
5
6  void pass_val ( int x , double &y ) ;
7  int main()
8  {
9      double y = 18.0 ;
10     int x = 3 ;
11     pass_val ( x , y );
12     cout << "\nx = " << x;
13     cout << "\ny = " << y ;
14 }
15 void pass_val ( int l , double &y)
16 {
17     l = l*2;  y = y+2;
18     cout << "\nl = " << l;
19     cout << "\ny = " << y ;
20 }
21
```

arguments

-x = passed by values
-y = pass by reference

parameters

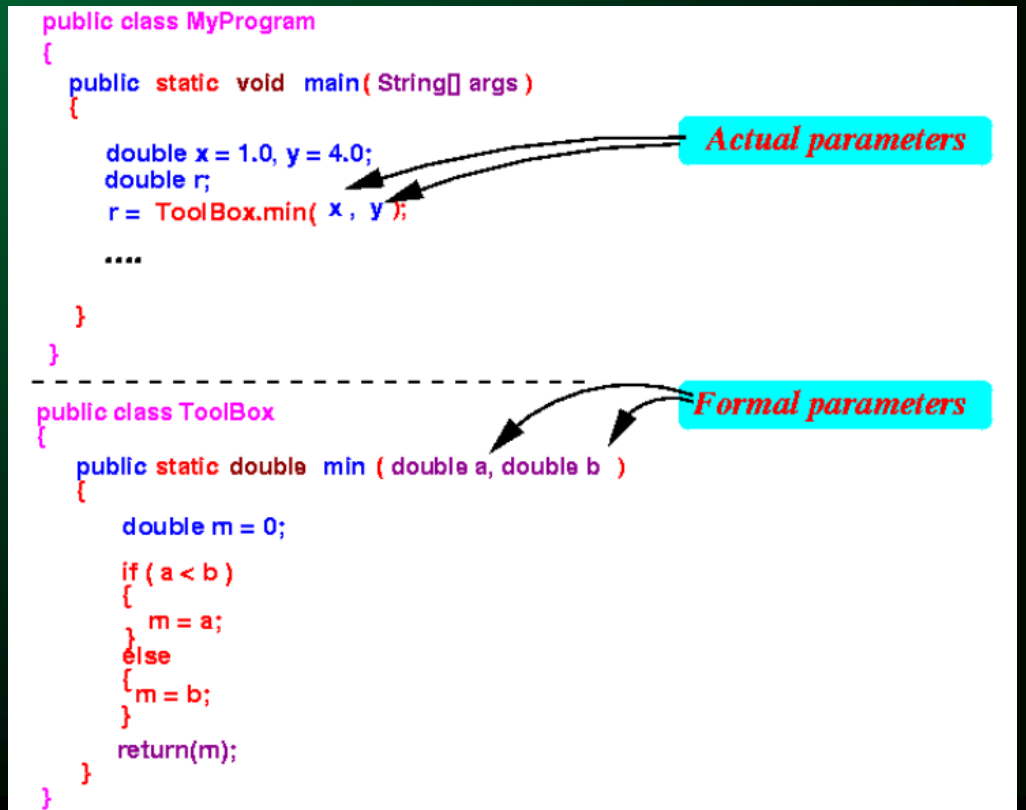
```
5  void pass_val ( int x , double &y ) ;
6  int main()
7  {
8      double y = 18.0 ;
9      int x = 3 ;
10     int i=100;
11     double j=91.1;
12     cout << "\nmain before -> x = " << x;
13     cout << "\nmain before  -> y = " << y ;
14
15     pass_val ( x  , y );
16     cout << "\nmain -> x = " << x;
17     cout << "\nmain -> y = " << y ;
18
19     cout << "\nmain before -> i = " << i;
20     cout << "\nmain before  -> j = " << j ;
21
22     pass_val ( i  , j );
23     cout << "\nmain -> i = " << i;
24     cout << "\nmain -> j = " << j ;
25
26 }
27 void pass_val ( int l , double &a)
28 {
29     cout << "\npass_val before -> l = " << l;
30     cout << "\npass_val before -> a = " << a ;
31
32     l = l*2;  a = a+2;
33     cout << "\npass_val -> l = " << l;
34     cout << "\npass_val -> a = " << a ;
35 }
36
```

7.2 Passing Data *Pass by value*

- The objective of design function with “pass by value” is the sending data from the caller to process in that function routine.

- The mechanism is the copying values of argument variables to be the initial value of parameter variables, so updating parameter variables in the calling function the argument value remain the same.

Parameter : formal parameters
Argument : actual parameters

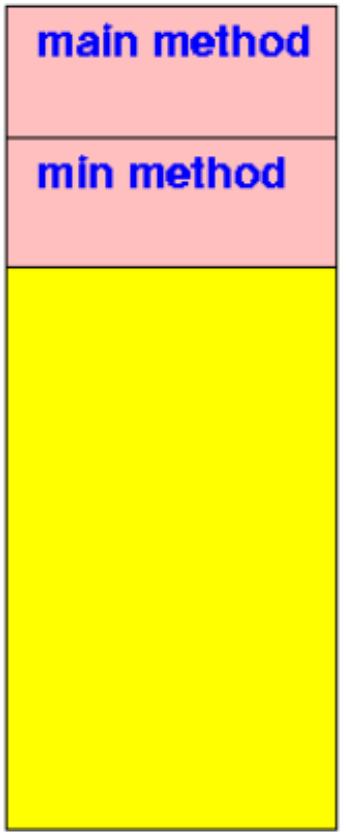


https://www.cs.emory.edu/~cheung/Courses/170/Syllabus/08/parameter2.html

ตย1. กลไกล pass by value

```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {
      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );

      ....

   }
}
```

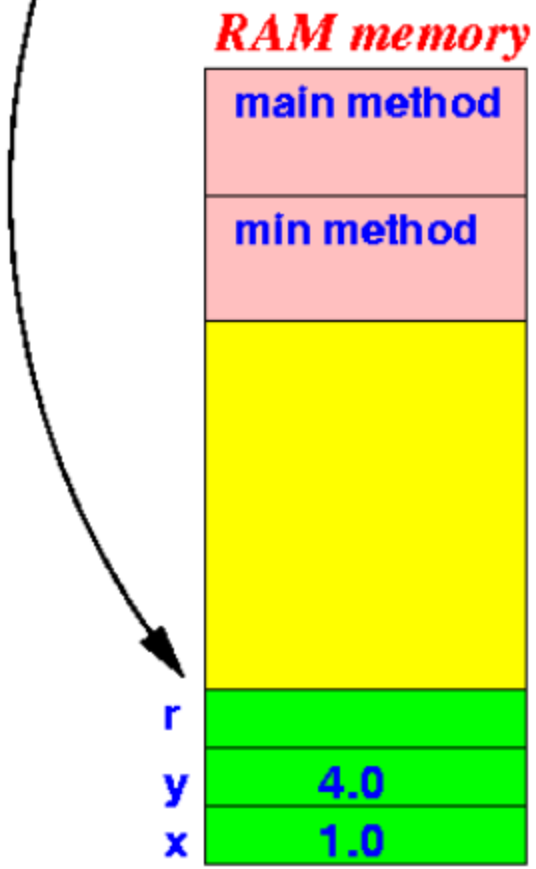
```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
         double m = 0;

         if ( a < b )
         {
            m = a;
         }
         else
         {
            m = b;
         }
         return(m);
    }
}
```



```
public class MyProgram
{
   public  static  void  main( String[] args )
   {

      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );

      ....

   }
}
```

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;

        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```

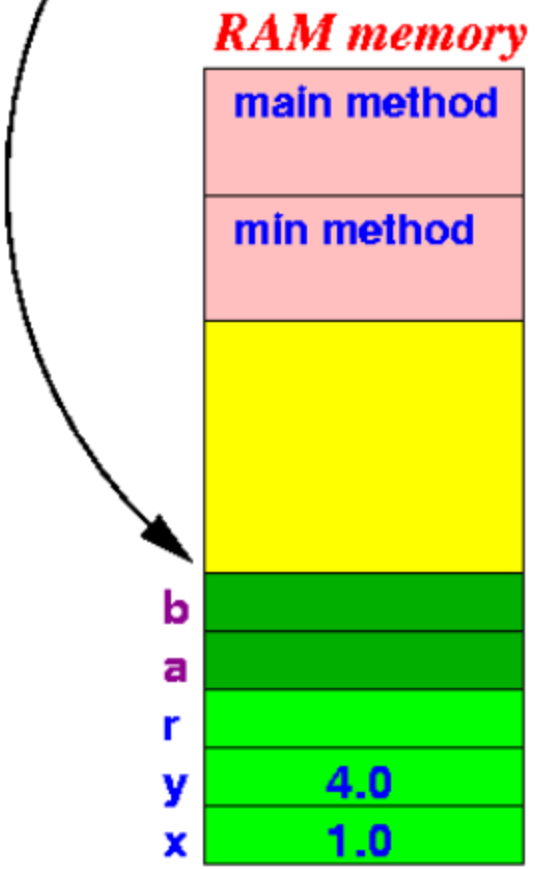
main creates its local variables



```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {

      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min(  x ,  y );

      ....

   }
}
```

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;

        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```

Pass-by-value:
1. create parameter variables

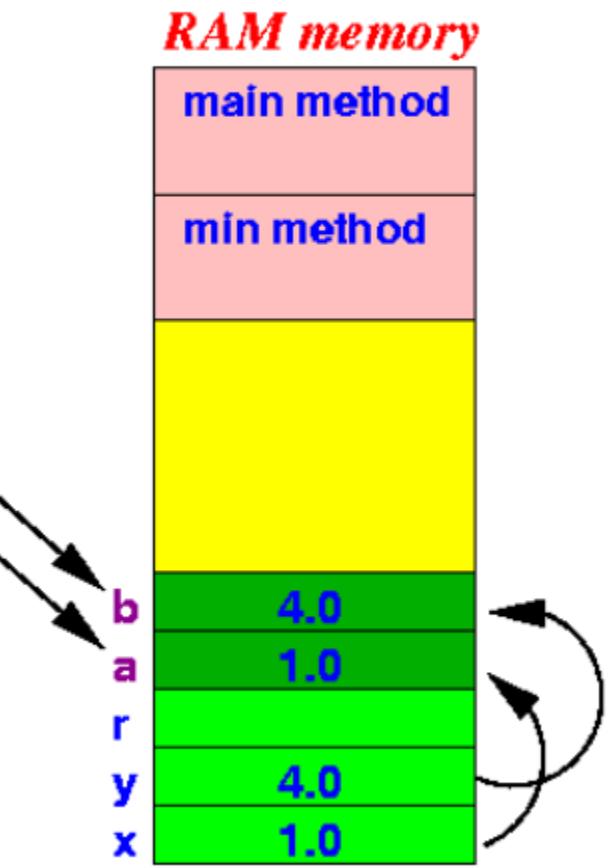


```
public class MyProgram
{
   public  static  void  main( String[] args )
   {
        double x = 1.0, y = 4.0;
        double r;
        r =  ToolBox.min( x , y );
        ....
   }
}
```

```
public class ToolBox
{
    public static double  min ( double a, double b )
    {
         double m = 0;
         if ( a < b )
         {
            m = a;
         }
         else
         {
            m = b;
         }
         return(m);
    }
}
```

Pass-by-value:

2. copy value of actual parameter to formal parameter



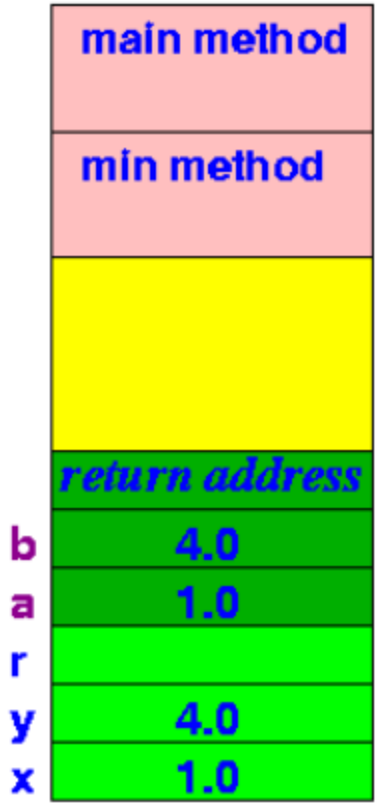
Method invocation mechanism: Jump to method

```
public class MyProgram
{
   public  static  void  main( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.min( x , y );
       ....
   }
}
```

return address

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;
        if ( a < b )
        {
          m = a;
        }
        else
        {
          m = b;
        }
        return(m);
    }
}
```

RAM memory



```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {

      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );
      ....

   }
}
```

Method invocation mechanism: Jump to method

return address

RAM memory

main method	
min method	
m	0
	return address
b	4.0
a	1.0
r	
y	4.0
x	1.0

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;

        if ( a < b )
        {
          m = a;
        }
        else
        {
          m = b;
        }
        return(m);
    }
}
```

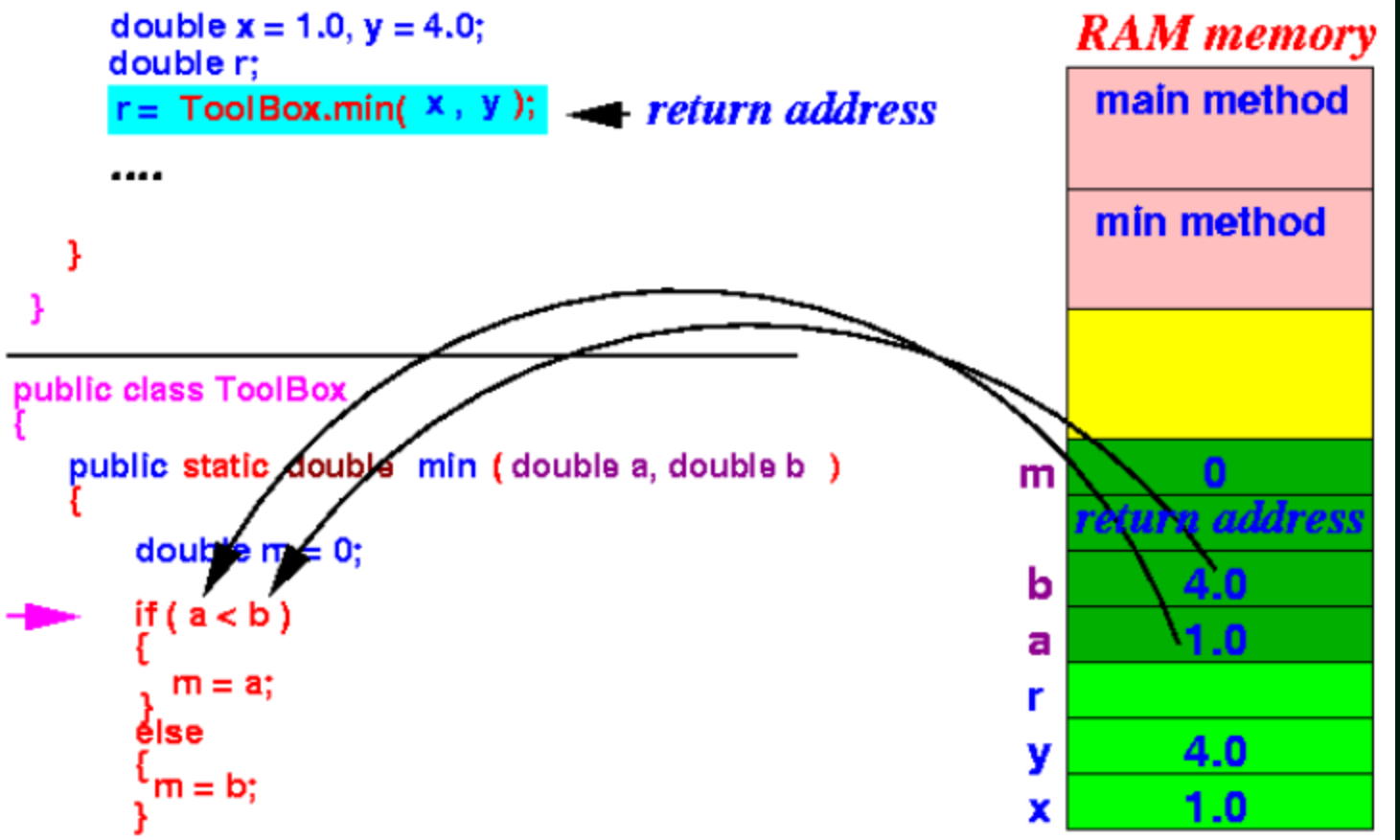
create local variable

```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.min( x , y );
       ....

   }
}
```

Method access information by:
Obtain information from parameter variable

return address

```
public class ToolBox
{
    public  static  double  min  ( double a, double b  )
    {
        double m = 0;
        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```



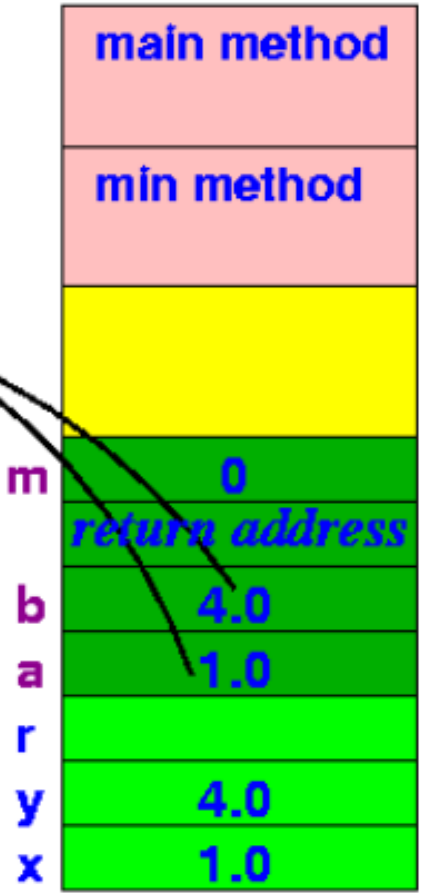
```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.min( x , y );
       ....
   }
}
```

return address

Method access information by: Obtain information from parameter variable

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;
        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```

RAM memory



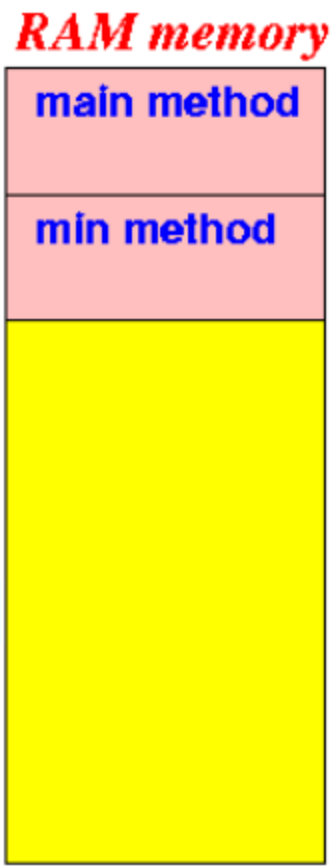
Ex2 :

```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.fun(  x ,  y );
       System.out.println(x);
       System.out.println(y);
       System.out.println(r);
   }
}
```

?

```
public class ToolBox
{
    public static double  fun  ( double a, double b  )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);
    }
}
```

Update parameter variable !

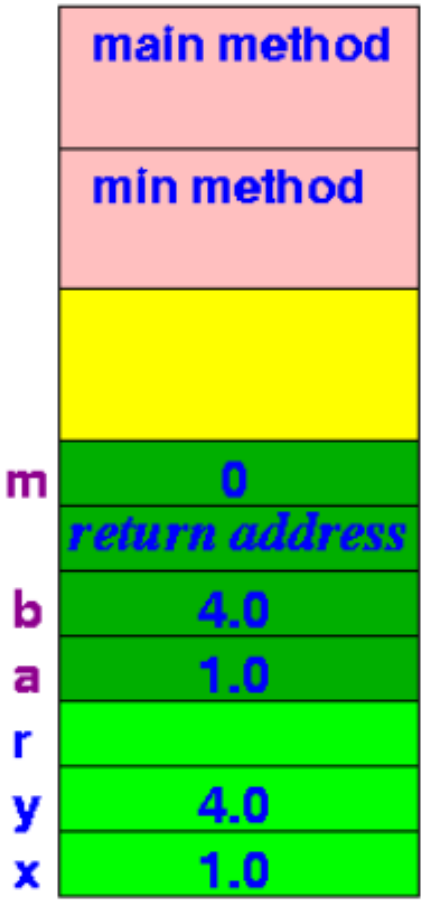


```
public class MyProgram
{
   public  static  void  main( String[] args )
   {
        double x = 1.0, y = 4.0;
        double r;
        r =  ToolBox.fun(  x ,  y );
        System.out.println(x);
        System.out.println(y);
        System.out.println(r);
   }
}
```

```
public class ToolBox
{
    public static double  fun  ( double a, double b  )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);
    }
}
```

Situation when method fun starts running

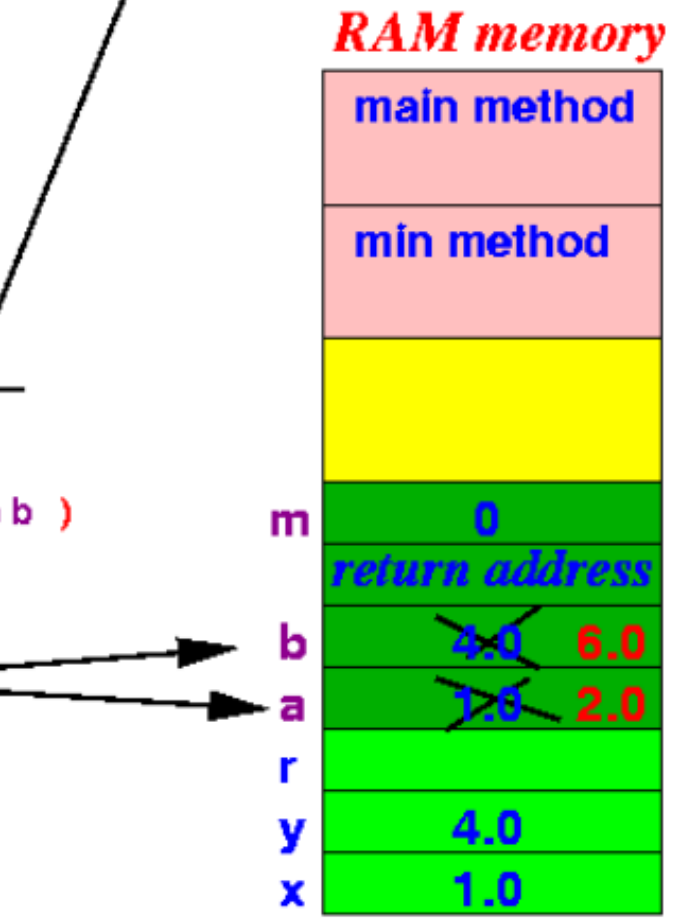
RAM memory



```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {

       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.fun(  x ,  y );

       System.out.println(x);
       System.out.println(y);
       System.out.println(r);
   }
}
```

```
public class ToolBox
{
    public static double  fun  ( double a, double b  )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);

    }
}
```

Assignment statements changes values stored in parameter var



- The *values* in the *actual parameters* (x and y) are unchanged !!!

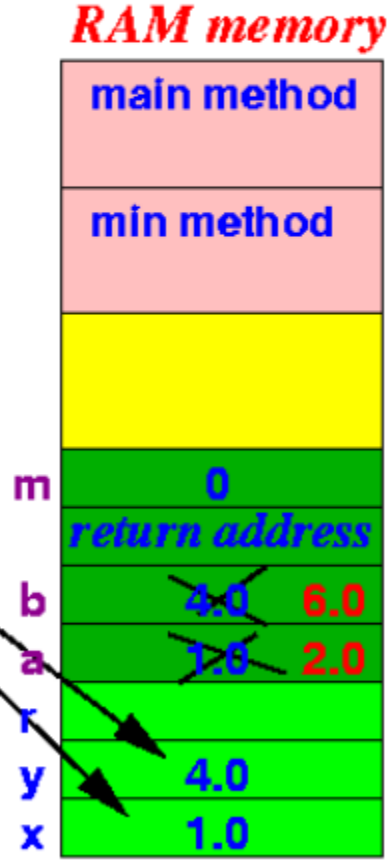
```
public class MyProgram
{
   public  static  void  main( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.fun(  x ,  y );
       System.out.println(x);
       System.out.println(y);
       System.out.println(r);
   }
}
```

```
public class ToolBox
{
    public static double  fun  ( double a, double b )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);
    }
```

That's why the **statements**

```
System.out.println(x);    ---> prints 1.0
System.out.println(y);    ---> prints 4.0
```

Actual parameters not affected !!



Passing by value in C/C++ :

main.cpp

```
1
2  #include <iostream>
3
4  using namespace std;
5
6  void pass_val ( int   , double  ) ;
7  int main()
8  {
9      double y = 18.0 ;
10     int x = 3 ;
11     pass_val ( x    , y    );
12     cout << "\nx = " << x;
13     cout << "\ny = " << y;
14 }
15 void pass_val ( int   a   , double    b     )
16 {
17      a = a*2;   b = b+2;
18     cout << "\na = " << a;
19     cout << "\nb = " << b;
20 }
21
22
```

X=?

Y =?

arguments

18.0

3

parameters

```
#include <iostream.h>

void print_value( int j);
int main()
{
  int i = 2;
  cout << "Value before = " << i << endl ;
  print_value(  i  );
  cout << "Value after  = " << i << endl ;
  return 0;
}

void print_value( int  j )
{
  cout << "value passed = " << j << endl ;
  j = j*2 ;
  cout << "value at end func = " << j << endl ;
}
```

```
C:\My Documents\sc107>passval1
Value before = 2
value passed = 2
value at end func = 4
Value after  = 2
```

What are the value of i , j after main function call print_value function ?

```
#include <iostream.h>
void print_value( int j);
int main()
{
  int i = 2;
  cout << "Value before = " << i << endl ;
  print_value(  i  );
  cout << "Value after  = " << i << endl ;
   i  = 40;
   print_value(  i  );
  cout << "Value after  = " << i << endl ;
  return 0;
}
void print_value( int  j )
{
  cout << "value passed = " << j << endl ;
  j = j*2 ;
  cout << "value at end func = " << j << endl ;
}
```

main.cpp

```cpp
#include <iostream>
using namespace std;

struct employee {
 char name[100];
 int age;
 float salary;
 char department[50];
};

// This function takes structure variable as parameter
void printEmployeeDetails(struct employee emp){
   cout << "\n*** Employee Details ***\n";
   cout << "Name : " << emp.name << "\nAge : "<< emp.age << "\nSalary : "
        << emp.salary << "\nDepartment : "<< emp.department;
}
int main(){
   struct employee manager ;

   printf("Enter Name, Age, Salary and Department of Employee\n");
   // Assigning data to members of structure variable
   cin >> manager.name >> manager.age >> manager.salary >> manager.department;

   // Passing structure variable to function
   printEmployeeDetails(manager);
}
```

```
Enter Name, Age, Salary and Department of Employee
apirak
30
1000.5
ram

*** Employee Details ***
Name : apirak
Age : 30
Salary : 1000.5
Department : ram
```

7.2 Passing Data

Passing by reference

Parameter : formal parameters
Argument : actual parameters

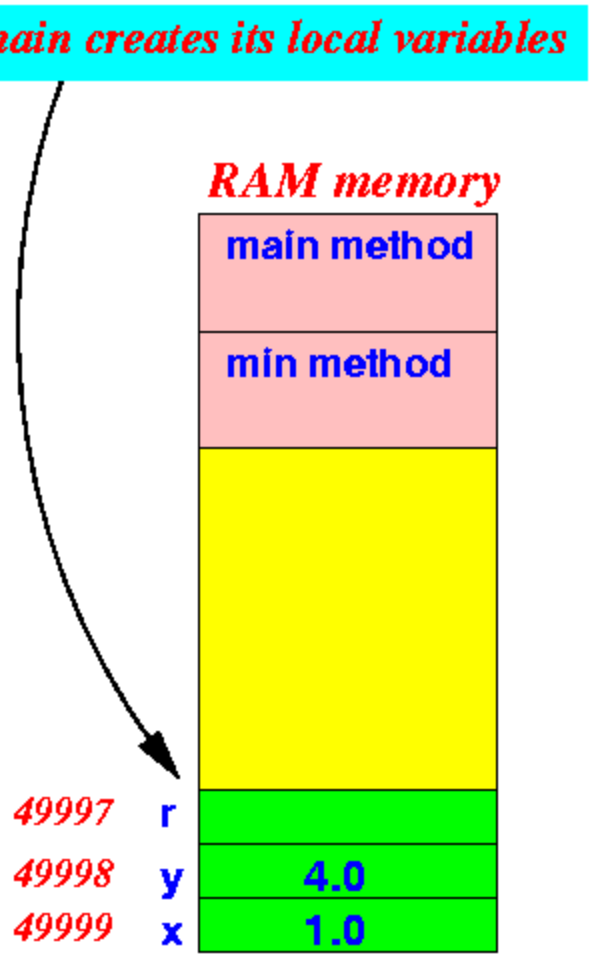
- The pass by reference is a two ways communication between the caller and its called function.
  - One for sending data from the caller to the function
  - One for receiving data of caller from its function
- However, the mainly objective of the method is getting information from other function.

Pass-by-reference mechanism

ตย2. กลไกล pass by reference

```
public class MyProgram
{
   public  static  void  main( String[] args )
   {
      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );
      ....
   }
}
```

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;
        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```

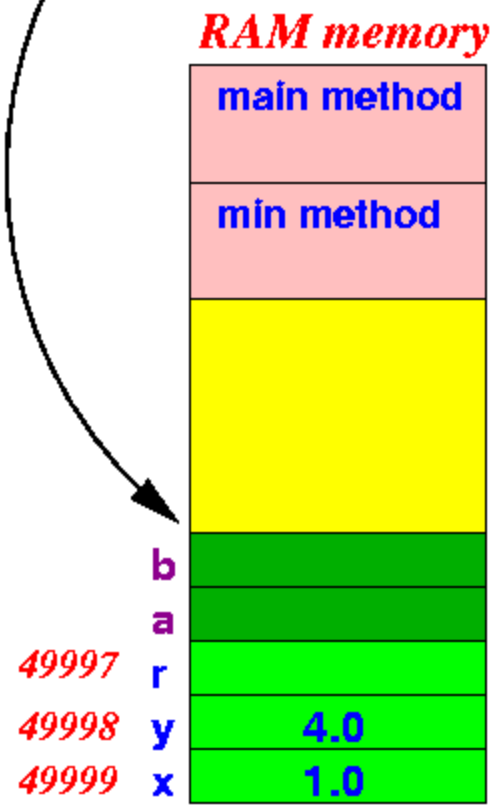


http://www.cs.emory.edu/~cheung/Courses/170/Syllabus/08/parameter3.html

```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {

      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min(  x ,  y );
      ....

   }
}
```

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;

        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```

Pass-by-reference:
1. create parameter variables

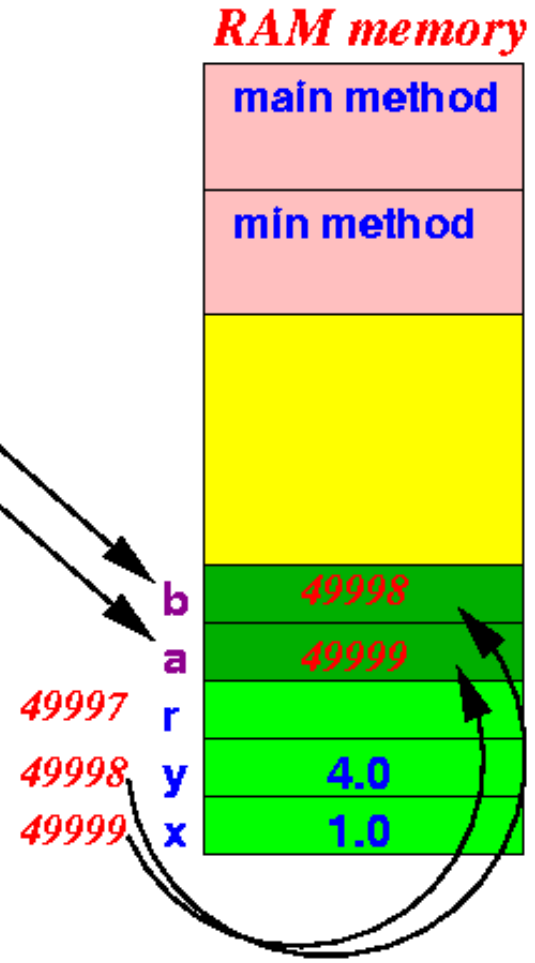


```
public class MyProgram
{
   public  static  void  main( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.min( x , y );
       ....
   }
}
```

```
public class ToolBox
{
    public static double  min ( double a, double b )
    {
        double m = 0;
        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```

Pass-by-reference:

2. copy reference of actual parameter to formal parameter

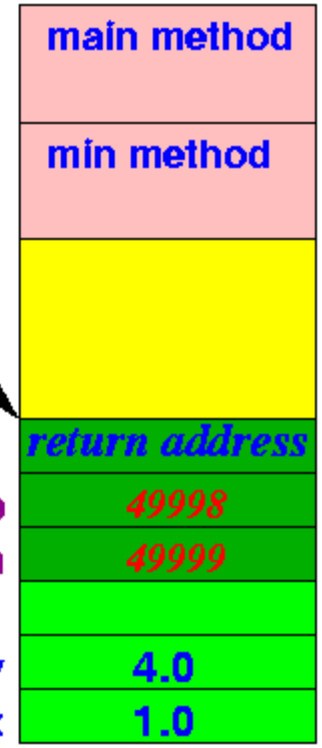


```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {

      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );
      ....

   }
}
```

Method invocation mechanism: Save return address

return address

RAM memory




```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;

        if ( a < b )
        {
          m = a;
        }
        else
        {
          m = b;
        }
       return(m);
    }
}
```

```
public class MyProgram
{
   public  static  void  main( String[] args )
   {

      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );
      ....

   }
}
```

Method invocation mechanism: Jump to method

RAM memory

	main method	
	min method	
m	0	
	return address	
b	49998	
a	49999	
49997	r	
49998	y	4.0
49999	x	1.0

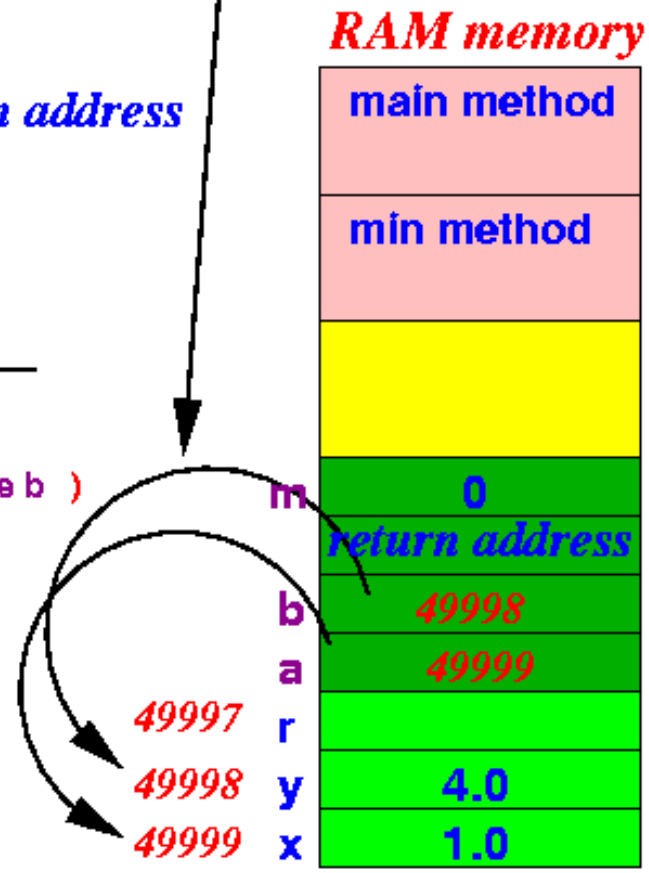
```
public class ToolBox
{
   public static double  min ( double a, double b  )
   {
      double m = 0;

      if ( a < b )
      {
         m = a;
      }
      else
      {
         m = b;
      }
      return(m);
   }
}
```

```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {
       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.min( x , y );
       ....
   }
}
```

return address

Method access information by:
1. Locate the actual parameter

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;
        if ( a < b )
        {
           m = a;
        }
        else
        {
           m = b;
        }
        return(m);
    }
}
```



```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {
      double x = 1.0, y = 4.0;
      double r;
      r =  ToolBox.min( x , y );
      ....
   }
}
```

return address

Method access information by:
2. Extract the information from the actual parameters

```
public class ToolBox
{
    public static double  min ( double a, double b  )
    {
        double m = 0;
        if ( a < b )
        {
          m = a;
        }
        else
        {
          m = b;
        }
        return(m);
    }
}
```

RAM memory

	main method	
	min method	
m	0	
	return address	
b	*49998*	
a	*49999*	
r		*49997*
y	4.0	*49998*
x	1.0	*49999*

Pass-by-reference mechanism update value case

```
public class MyProgram
{
   public  static  void  main( String[] args )
   {

       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.fun(  x ,  y );

       System.out.println(x);
       System.out.println(y);
       System.out.println(r);
   }
}
```

```
public class ToolBox
{
    public static double  fun  ( double a, double b  )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);

    }
}
```

Situation when method fun starts running

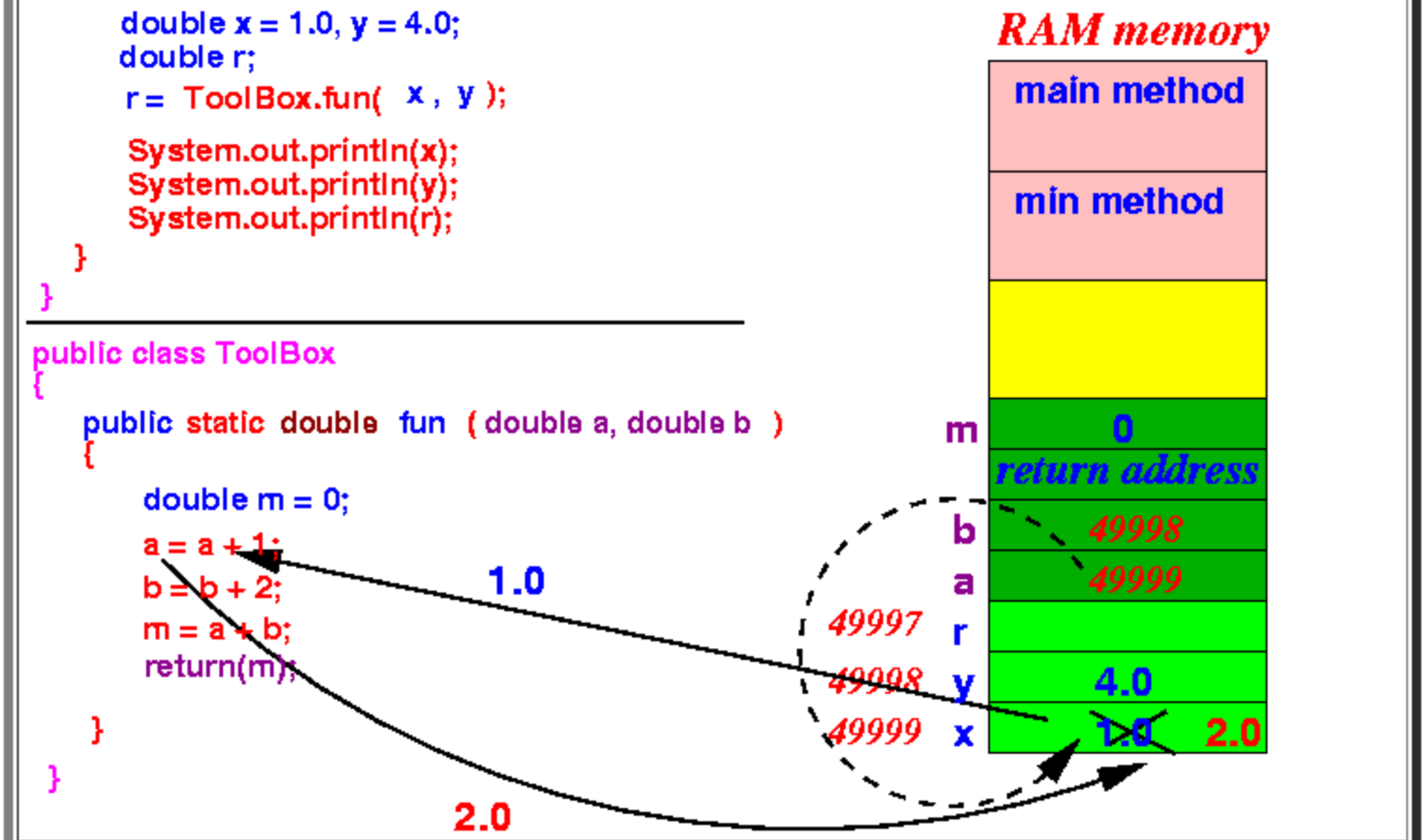
RAM memory

		main method
		min method
	m	0
		return address
	b	49998
	a	49999
49997	r	
49998	y	4.0
49999	x	1.0

```
public class MyProgram
{
   public  static  void  main ( String[] args )
   {

       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.fun(  x ,  y );

       System.out.println(x);
       System.out.println(y);
       System.out.println(r);
   }
}
```

```
public class ToolBox
{
    public static double  fun  ( double a, double b  )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);

    }
}
```

Assignment statements changes values stored in actual parameter



Example of pass by reference

main.cpp

```cpp
#include <stdio.h>

void swapnum(int &i, int &j) {
  int temp = i;
  i = j;
  j = temp;
}

int main(void) {
  int a = 10;
  int b = 20;

  printf("before : A is %d and B is %d\n", a, b);
  swapnum(a, b);
  printf("after : A is %d and B is %d\n", a, b);
  return 0;
}
```

```
before : A is 10 and B is 20
after : A is 20 and B is 10


...Program finished with exit code 0
```

7.2 Passing Data

Passing by reference

```
void get_values ( float &income , float &expense );

void get_values ( float &income , float &expense )
{
    cout << ” Enter income  amount " ;
    cin  >> income ;
    cout << ” Enter expense amount " ;
    cin  >> expense ;
}
```

C/C++ Pass by reference

Parameter : formal parameters
Argument : actual parameters

- Support all data types using " & " on actual parameter
- By default ,passing "Array data type" is pass by reference
- The formal parameters of the caller must be "variable" only.

```
#include <iostream.h>
void get_value( int &j );
int main()
{
  int i = 0 ;
  cout << "before call i = " << i << endl ;
  get_value( 10 ) ;
  cout << "after call i  = " << i << endl ;
  return 0;
}

void get_value( int &j )
{
  cout << "value passed to func = " << j << endl ;
  cin >> j ;
  cout << "value at end func  = " << j << endl ;
}
```

!! Compiler error

```
8
9   #include <iostream>
10  using namespace std;
11  struct myStructure {
12      int x;
13      int y;
14  };
15
16  void swap(myStructure &ms)
17  {
18      int temp = ms.x;
19      ms.x = ms.y;
20      ms.y = temp;
21  }
22
23  int main()
24  {
25      myStructure ms = { 50, 60 };
26
27      cout<<"\n\nBefore swap, inside main():\n";
28      cout<<"x = "<<ms.x<<"\ty = "<<ms.y;
29
30      swap(ms);
31
32      cout<<"\n\nAfter swap, inside main():\n";
33      cout<<"x = "<<ms.x<<"\ty = "<<ms.y;
34      cout<<endl;
35      return 0;
36  }
```

```
Before swap, inside main():
x = 50   y = 60

After swap, inside main():
x = 60   y = 50
```

```
#include <iostream.h>
void get_value( int &j );
int main()
{
  int i = 0 ;
  cout << "before call i = " << i << endl ;
  get_value( i ) ;
  cout << "after call i  = " << i << endl ;
  return 0;
}

void get_value( int &j )
{
  cout << "value passed to func = " << j << endl ;
  cin >> j ;
  cout << "value at end func  = " << j << endl ;
}
```

```
before call i = 0
value passed to func = 0
13
value at end func  = 13
after call i  = 13
```

What the value of i , j after main function call j2Mul function are ?

```
#include <iostream.h>
void j2Mul ( int &j );
int main()
{
  int i = 1 ;
  cout << "before call i = " << i << endl ;
   j2Mul( i ) ;
  cout << "after call i  = " << i << endl ;
   j2Mul( i ) ;
  cout << "after call i  = " << i << endl ;
  return 0;
}

void j2Mul ( int &j )
{
  cout << "value at being func  = " << j << endl ;
   j  = j*20;
  cout << "value at end func  = " << j << endl ;
}
```

Array & Function in C/C++

- “pass by reference”

Pass array to function (pass by reference)

main.cpp

```cpp
using namespace std;
void updateArray( int data[5] )
{
    for(int i = 0 ; i < 5 ; i++) {
        data[i] = data[i]+1000;
    }
}
void showarray( int data[5] )
{
    for(int i = 0 ; i < 5 ; i++) {
        cout << data[i] << ",";
    }
}
int main() {

    int Roll_Number[5] = {100,101,102,103,104};

    cout << "\nbefore update \n";
    showarray(Roll_Number);
    updateArray(Roll_Number);
    cout << "\nafter update \n";
    showarray(Roll_Number);
}
```

```
before update
100,101,102,103,104,
after update
1100,1101,1102,103,104,
```

Pass array to function (pass by reference)

```cpp
#include <iostream>
using namespace std;
void updateArray( int data[ ],int n )
{
    for(int i = 0 ; i < n ; i++) {
        data[i] = data[i]+1000;
    }
}
void showarray( int data[ ],int n )
{
    for(int i = 0 ; i < n ; i++) {
        cout << data[i] << ",";
    }
}
int main() {

    int Roll_Number1[5] = {100,101,102,103,104};
    int Roll_Number2[5] = {1000,1010,1020};

    cout << "\nbefore update Roll_Number1 \n";
    showarray(Roll_Number1,5);
    updateArray(Roll_Number1,3);
    cout << "\nafter update \n";
    showarray(Roll_Number1,5);

    cout << "\nshow  Roll_Number2\n";
    showarray(Roll_Number2,3);
}
```

Not allow to return Array

main.cpp

```cpp
#include <iostream>
using namespace std;
//return pointer data type
int * returnArray(  )
{
     int myarray[100];
     return myarray;
}
//return pointer data type
int *returnPassArray( int passData[ ] )
{
    return passData;
}
int main() {

    int myData1[100];
    myData1 = returnArray();

    int myData2[100];
    myData2 = returnPassArray( myData1);

}
```

Pass array of characters or string

main.cpp

```cpp
//C++
//#include <iostream>
//using namespace std;

//TurboC
#include <iostream.h>

void showarray( char *data )
{
    cout << "data -> " << data;
    cout << "\n";
}
void updateArray( char data[ ] )
{
   showarray(data);
     for(int i = 0 ; data[i] != '\0' ; i++) {
        data[i] = data[i]+1;
     }
     showarray(data);
}

int main() {
    //array of charector or string
    char mystring1[] = "012345678";
    updateArray(mystring1);

    //string constant
    char *mystring2 = "abcdef";
    updateArray(mystring2);
    cout << "end ";
}
```

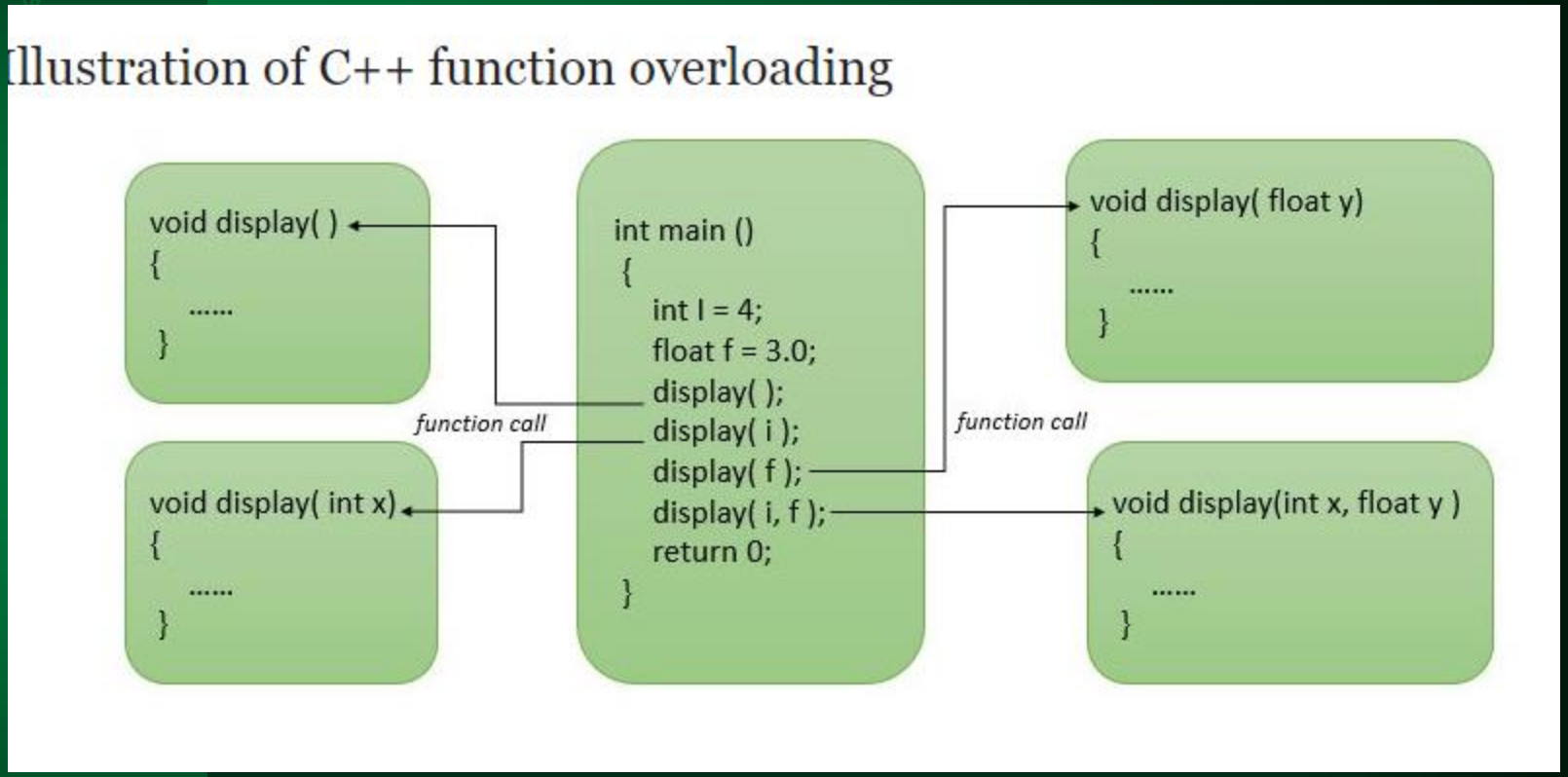
Overload function

improve readability

main.cpp

```cpp
#include <iostream>
using namespace std;

void print_int(int i) {
  cout << " Here is int " << i << endl;
}
void print_double(double  f) {
  cout << " Here is float " << f << endl;
}
void print_array(char *c) {
  cout << " Here is char* " << c << endl;
}

int main() {
  print_int(10);
  print_double(10.10);
  print_array("ten");
  return 0;
}
```

Overload function



main.cpp

```cpp
#include <iostream>
using namespace std;

void print(int i) {
  cout << " Here is int " << i << endl;
}
void print(double  f) {
  cout << " Here is float " << f << endl;
}
void print(char *c) {
  cout << " Here is char* " << c << endl;
}

int main() {
  print(10);
  print(10.10);
  print("ten");
  return 0;
}
```

Apirak :

More about Function Prototypes

- A function prototypes consists of
  - The function's return type.
  - Function Name
  - argument list
- However, variable's name of argument list may ignore.

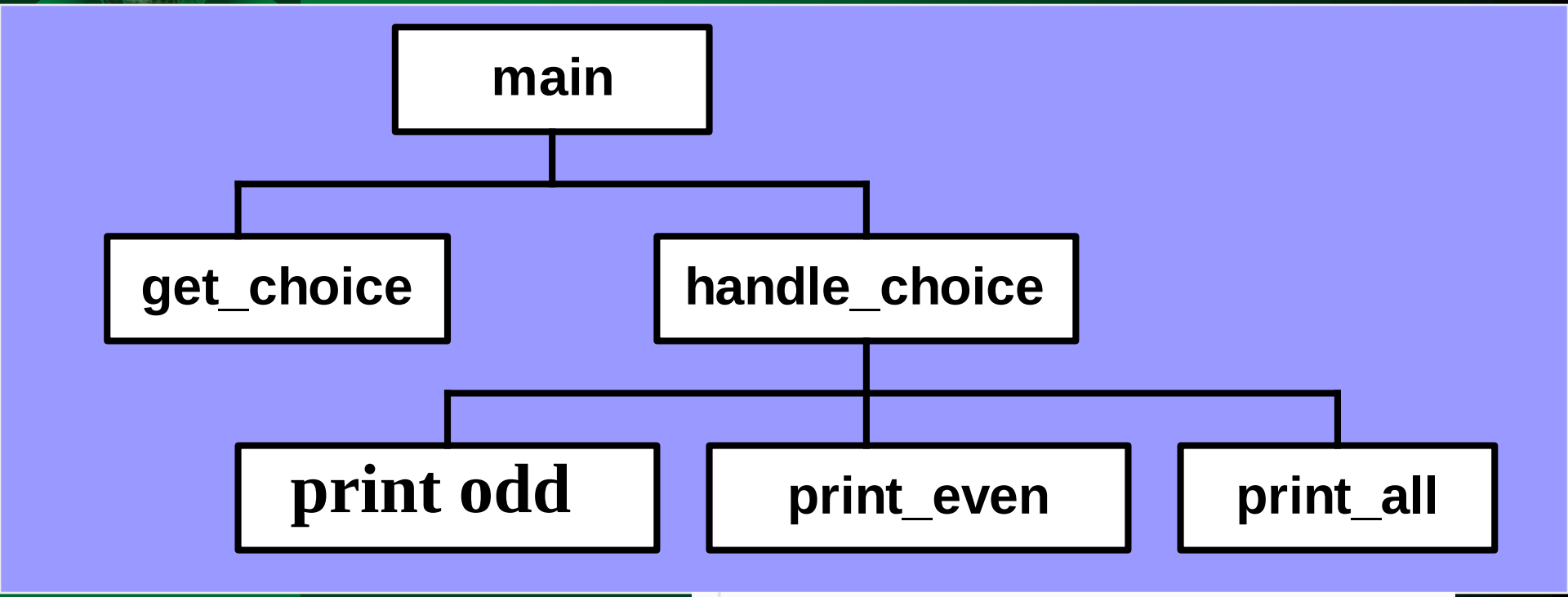
NOTE :More about Function Prototypes in C/C++

variable's name of argument list may ignore.

```
//double celsius_to_fahrenheit( double celsius ) ;
double celsius_to_fahrenheit( double ) ;

double celsius_to_fahrenheit( double celsius )
{
        double fahr ;
        fahr = celsius * ( 9.0/5.0) + 32.0 ;
        return  fahr ;
}
```

```
//void Swap (  int &x ,  int &y ) ;
void Swap (  int &  ,  int &  ) ;
void Swap (  int &x ,int &y )
{
   int temp ;
   temp = x ;
   x = y ;
   y = temp ;
}
```

Rewrite function using pass by reference




```
/tmp/MmMbJeNdMr.o
Which series do you wish to display?
1 - Odd numbers from 1 to 30
2 - Even numbers from 1 to 30
3 - All numbers from 1 to 30
```

Previous version

```cpp
#include <iostream>
using namespace std;

int getchoice() {
 int choice;  // variable for user input
 do // loop until a valid choice is entered
  {
    cout << "Which series do you wish to display?\n";
    cout << "1 - Odd numbers from 1 to 30\n";
    cout << "2 - Even numbers from 1 to 30\n";
    cout << "3 - All numbers from 1 to 30\n";
    cin >> choice;  // get choice from user
    if ((choice < 1) || (choice > 3))
     {  // if invalid entry, give message
        cout << "Choice must be 1, 2, or 3\n";
     }
  } while ((choice < 1) || (choice > 3));
  return choice;
}

void printOdd() {
   for (int i = 1; i <= 30; i = i + 2)
        cout << i << ' ';
   cout << endl;
}

void printEvent() {
   for (int i = 2; i <= 30; i = i + 2)
        cout << i << ' ';
   cout << endl;
}
```

```cpp
void printAll() {
   for (int i = 2; i <= 30; i = i + 2)
        cout << i << ' ';
   cout << endl;
}

int main()
{

    int choice = getchoice();
    switch (choice)
    {
     case 1:
        printOdd();
       break;
     case 2:
        printEvent() ;
       break;
     case 3:
        printAll( );
       break;
    }
  return 0;
}
```

Rewrite program with procedural concept

แบบฝึกหัด 2

จงเขียนโปรแกรมต้องการนำระเบียนลูกค้า(customer) ของสหกรณ์ออมทรัพย์รามคำแหง ที่รองรับการจัดเก็บข้อมูลไม่เกิน 500 คน ระเบียนลูกค้าประกอบด้วย รหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก (Deposit)

Interest = deposit*10%;
Deposist = Deposit+interst

Menu

1. Open a new Account
2. Deposit
3. Withdraw
4. Compound interest
5. Quit

Please select 1 / 2 / 3 / 4 / 5 =?

main
Opennewacc
widthdraw
report
menu
deposit
Compond interest

```
int main()
{
    struct person customer[500];
    int n=0;
    int choice;
    char id[5];
    float _deposit,_widthdraw;
    char name[100];
    char addrs[100];
    int res;
    choice = displaymenu();
    while(choice!=6) {
        switch(choice){
            case 1:
            /*
            int opennewacc(char id[5],
                float deposit,char name[], char addrs[]
                int &n,struct person p[])
            */
                        cout << "id : ";
                        cin >> id;
                        cout << "name : ";
                        cin >> name;
                        cout << "addrs : ";
                        cin >> addrs;
                        cout << "deposit : ";
                        cin >> _deposit;
                        res =opennewacc(id,
                                        _deposit,
                                        name,
                                        addrs,
                                        n,
                                        customer);
                        if(res==0) {
                            cout << "accout duplicate error" << endl;
                        }
                        break;
```

```
1506            case 2:
1507            /*
1508                int deposit(char id[5],float m,int n,struct person p[])
1509            */
1510                    cout << "id : ";
1511                    cin >> id;
1512                    cout << "deposit : ";
1513                    cin >> _deposit;
1514                    res = deposit(id,_deposit,n,customer);
1515
1516                    break;
1517            case 3:
1518            /*
1519            int widthdraw(char id[5],float m,int n,struct person p[])
1520            */
1521                    cout << "id : ";
1522                    cin >> id;
1523                    cout << "widthdraw : ";
1524                    cin >> _widthdraw;
1525                    res = widthdraw(id,_widthdraw,n,customer);
1526                    break;
1527            case 4:
1528                    break;
1529
1530            case 5:
1531                    report( n,customer );
1532                    break;
1533        }
1534        choice = displaymenu();
1535    }
1536
1537    return 0;
1538 }
1539
```

```
1366  #include <iostream>
1367  #include <string.h>
1368  #include <iomanip>
1369
1370  using namespace std;
1371
1372  struct person {
1373    char id[5];
1374    char name[30];
1375    char addrs[50];
1376    float deposit;
1377  };
1378  int displaymenu()
1379  {
1380      int choice;
1381      do {
1382      cout << "1.open acc" << endl;
1383      cout << "2.deposit acc" << endl;
1384      cout << "3.widthdraw acc" << endl;
1385      cout << "4.compond interest acc" << endl;
1386      cout << "5.reports " << endl;
1387      cout << "6.quit" << endl;
1388      cin >> choice;
1389      }while( (choice >=1 && choice << 5) == false );
1390      return choice;
1391  }
```

```
1393  int checkaccExist(char id[5],int n,struct person p[])
1394  { //0..n-1 found,-1=not found
1395      for(int i=0; i< n ; i++) {
1396          if( strcmp( id, p[i].id)==0) {
1397              return i;
1398          }
1399      }
1400      return -1;
1401  }
1402  int opennewacc(char id[5],
1403     float deposit,char name[], char addrs[],
1404     int &n,struct person p[])
1405  {
1406      int res = checkaccExist(id,n,p); //0..n-1 found,-1=not found
1407      if( res >=0) {
1408          return 0;
1409      }
1410      /*
1411  struct person {
1412    char id[5];
1413    char name[30];
1414    char addrs[50];
1415    float deposit;
1416  }
1417      */
1418      strcpy( p[n].id, id );
1419      strcpy( p[n].name, name );
1420      strcpy( p[n].addrs, addrs );
1421      p[n].deposit = deposit;
1422      n++;
1423      return 1;
1424  //1 success
1425  //0 not success
1426  }
1427  int deposit(char id[5],float m,int n,struct person p[]) {
1428  //1 success
1429  //0 not success
1430      int idex = checkaccExist(id,n,p); //0..n-1 found,-1=not found
1431      if( idex < 0) {
1432          return 0;
1433      }
1434      p[idex].deposit += m;
1435  return 0;
1436  }
```

```
1438  int widthdraw(char id[5],float m,int n,struct person p[]) {
1439  //1 success
1440  //0 not success
1441      int idex = checkaccExist(id,n,p); //0..n-1 found,-1=not found
1442      if( idex < 0) {
1443          return 0;
1444      }
1445      if(p[idex].deposit -m < 100 ) {
1446          return 0;
1447      }
1448      p[idex].deposit -= m;
1449      return 1;
1450  }
1451  void compondinterest( float rate,int n,struct person p[]) {
1452  }
1453
1454  void report( int n,struct person p[]) {
1455      for(int i = 0; i < n ; i++) {
1456          cout << "id : " << setw(10) ;
1457          cout << p[i].id << setw(10) ;
1458          cout << "name : " << setw(10) ;
1459          cout << p[i].name << setw(10) ;
1460          cout << "addrs : " << setw(10) ;
1461          cout << p[i].addrs << setw(10) ;
1462          cout << "deposit : " << setw(10) ;
1463          cout << p[i].deposit<< setw(10) ;
1464          cout << endl;
1465      }
1466  }
```

ตัวอย่าง

ให้นักศึกษาเขียนโปรแกรมขายสินค้าและพิมพ์ใบเสร็จรับเงิน
โดยข้อมูลในระบบมีไม่เกิน **100** รายการ
ข้อมูลขายสินค้าด้วย

- รหัสสินค้า(id) เช่น “AB001”
- ชื่อสินค้า(name) เช่น “CAFÉ”
- ราคาขาย(cost) เช่น 30.6
- จำนวนสินค้าในสต๊อก (instock) เช่น 5

Rewrite your code with Function approach

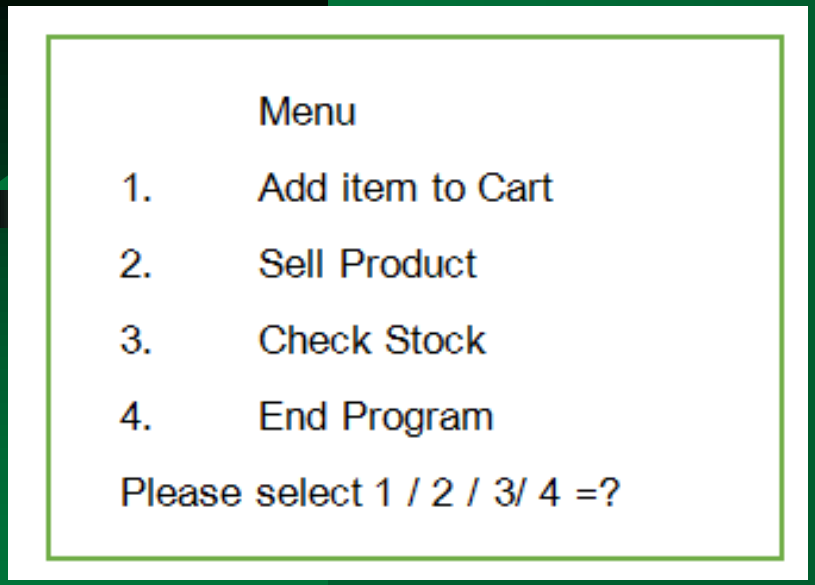
BillStructure.cpp

Rewrite the code using function approach

ตัวอย่าง

ให้นักศึกษาเขียนโปรแกรมขายสินค้าและพิมพ์ใบเสร็จรับเงิน
โดยข้อมูลในระบบมีไม่เกิน 100 รายการ
ข้อมูลขายสินค้าด้วย

- รหัสสินค้า(id) เช่น “AB001”
- ชื่อสินค้า(name) เช่น “CAFÉ”
- ราคาขาย(cost) เช่น 30.6
- จำนวนสินค้าในสต๊อก (instock) เช่น 5

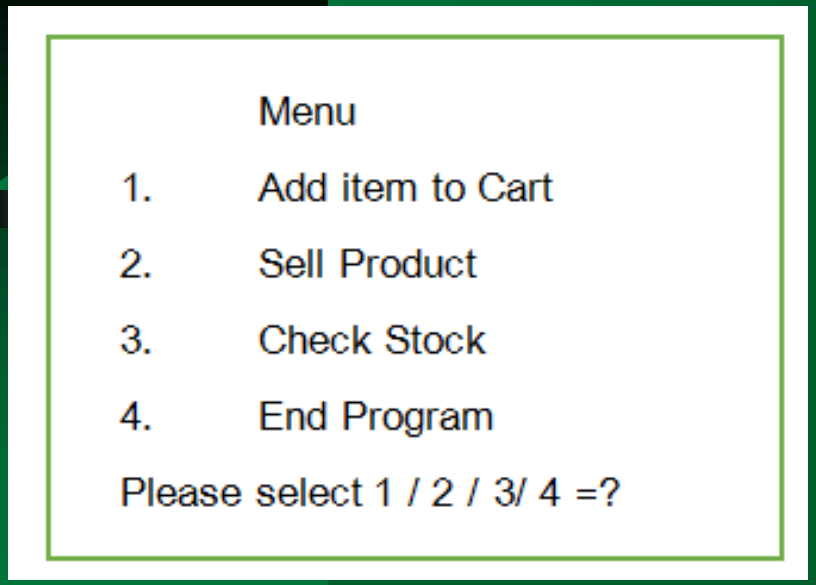


กด 1

Input your product data:
id : AB001
name : CAFE
Cost : 30.6
Stock : 5

กด 2

Input your product id: AB001
Input your sell volume : 2
name : CAFE
Cost : 30.6
stock : 3
Pay : 61.2 bath



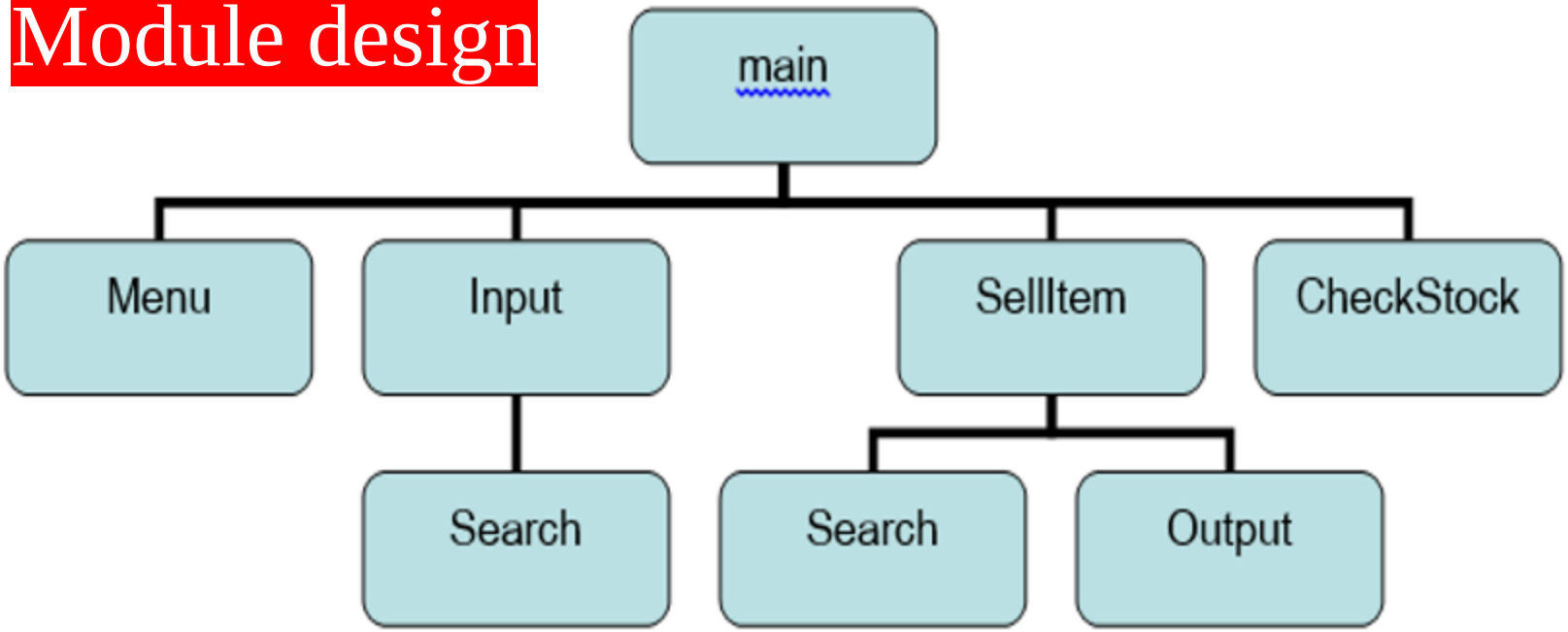
กด 3

```
product List:
Id          name    cost  stock
- - - - - - - - - - - - - - - - - -
AB001  CAFÉ   30.6    3
```

กด 4

exit from program

Module design



EX1 จากตัวอย่างข้างต้นให้ นักศึกษาปรับโปรแกรม โดยให้มีการตรวจดังนี้

กรณี กด 1 : ให้ตรวจสอบว่ารหัสที่ป้อน มีซ้ำหรือไม่ ถ้าไม่มีจึงจะอนุญาตให้ผู้ใช้ป้อนข้อมูล

กรณี กด 2: ให้ตรวจสอบว่ารหัสสินค้าที่ผู้ใช้ป้อนมีอยู่ในระบบหรือไม่ กรณีไม่พบให้แสดงข้อความ “product not exist”

กรณี กด 3 ให้แสดงรายการทั้งหมด และแสดงยอดรวมของ cost

7.3 Using Library Functions

- C++ provide a lot of standard functions,called *library functions for programmer*.

- The prototypes of library functions are defined to your program using #include compiler directive.

```
#include <iostream.h>
#include <math.h>

int main(void)
{
   double x,y ,z ;
   cout << "get x= "; cin >> x ;
   cout << "get y= "; cin >> y ;
   z = pow( x, y );
   cout << "get x**y = " << z ;

   return 0;
}
```

Popular Math Function

Function	Prototype	Description
abs	int abs(int x)	return absolute value of an int
labs	long abs(long int x)	return absolute value of long int
fabs	double abs(double x)	return absolute value of double
ceil	double ceil(double x)	rounds up to a whole number
floor	double floor(double x)	rounds down to a whole number
pow	double pow(double x,double y)	calculate x to the power of y
pow10	double pow10(int x)	caculate 10 to the power of y
sqrt	double sqrt(double x)	caculate the positive square root of x

Functions for Working with character

- C++ also provide many functions for analyzing and changing characters.

Function	Prototype	Description
isupper	int isupper(char x)	Determines if a character is upper
islower	int islower(char x)	Determines if a character is lower
isalpha	int isalpha(char x)	Determines if a character is ‘a’-’z’
isdigit	int isdigit(char x)	Determines if a character is ‘0’-’9’
toupper	int toupper(char x)	Converts a character to upper
tolower	int tolower(char x)	Converts a character to lower

Run | Debug | Stop | Share

main.cpp

```cpp
#include <iostream>
using namespace std;
#include <ctype.h>
int main()
{
 char c;
 cout << "enter a characer :";
 cin >>c;
 if( isalpha(c) )
  {
    if( isupper(c) )
     { cout << c << " is an upper leter\n"; }
    else
     { cout << c << " is an lower leter\n"; }
  }
  if( isdigit(c) )
    { cout << c << " is a number\n"; }
  if( !( isdigit(c) || isalpha(c) ) )
    { cout << c << " is a number\n"; }
}

```

```
C:\My Documents\sc107>charfun
enter a characer :a
a is an lower leter

C:\My Documents\sc107>charfun
enter a characer :1
1 is a number
```

Design methodology

- ***Top-down design**: begins with the function at top of diagram and works toward the function at the bottom of the diagram.*

- ***Button-up design**: involves begins with the bottom of diagram and working your way up.*

ตัวอย่าง (3)

จงเขียนโปรแกรมอ่านข้อมูล 2 ชนิด คือ A และ B ซึ่งชนิดละ20 จำนวน จากนั้นให้คำนวณค่า

A_bar = Sum i = 1 .. 20 ของ A

B_bar = Sum i = 1 .. 20 ของ B

$$Corr(A, B) = \frac{n*Sum(AiBi) - Sum(Ai)*Sum(Bi)}{sqrt(n*(sum(Ai*Ai) - Sum(Ai)*Sum(Ai)) * n*(sum(Bi*Bi) - Sum(Bi)*Sum(Bi)))}$$

กำหนดให้ z = sqrt(x) เป็นbuild in function ของการหาค่า root x

```
4
5  void GetData ( float Z[  ] ,int n );
6  float  Sum ( float Z[ ] , int n ) ;
7  void  MultAry ( float W[ ] , float X[ ] , float Z[ ] , int n ) ;
8
9
10 int main( )
11 {
12     float  A [ 20 ] , B[ 20 ] ,T[20] ;
13     float  A_bar , B_bar ,corrAB , X , Y;
14     float  AB_bar , AA_bar , BB_bar ;
15    cout << " input DATA of  Matrix A \n"  ;
16    GetData ( A ,20 ) ;
17    cout << " input DATA of  Matrix B \n"  ;
18    GetData ( B , 20  ) ;
19     A_bar = Sum ( A  , 20 ) ;
20     B_bar   = Sum ( B , 20 );
21     MultAry( A , A , T , 20 )  ; AA_bar = Sum ( T, 20 ) ;
22     MultAry(  B , B , T , 20 )  ; BB_bar = Sum ( T, 20 ) ;
23     MultAry( A , B , T , 20 )  ; AB_bar = Sum ( T, 20 ) ;
24
25     X   =  (20*AB_bar - A_bar*B_bar) ;
26     Y   = (  ( 20* (AA_bar - A_bar* A_bar) ) *
27                ( 20* (BB_bar - B_bar* B_bar) ) );
28     corrAB = X / sqrt ( Y ) ;
29
30     cout << "\ncorr(A,B)"  <<  corrAB ;
31
32     return 0;
33 }
34
```

Apirak :

สมมุติว่า Apartment แห่งหนึ่ง มีจำนวนห้องพัก 100 ห้อง จงเขียนโปรแกรมคำนวณค่าปริมาณการใช้น้ำดังนี้

1. โปรแกรมแสดงเมนู
    1. add meter
    2.Caculate meter
    3.Diplay meter
2. ข้อมูลที่รับเข้าประกอบด้วย

1.เลขประจำมิเตอร์ (no. of meter) จำนวน 4 ตัวอักษร เช่น A001,A002

2. ค่าม (start meter value)

3.ค่ามิเตอร์

Best practice : Guide for coding subroutine

- Get single value
- Pass value and get value
- Return multiple value using pass by reference
- return local array

Return function vs pass by ref

```cpp
#include <iostream>

using namespace std;

int getonedata() {
    int i;
    cout << " enter data : " ;
    cin>> i;
    return i;
}
int main()
{
    int x = getonedata();
    return 0;
}
```

```cpp

#include <iostream>

using namespace std;

void getonedata(int &i) {
    cout << " enter data : " ;
    cin>> i;
}
int main()
{
    int x;
    getonedata(x);
    return 0;
}

```

Return function vs pass by ref

```cpp
#include <iostream>
using namespace std;

int sumx() {
    return 10;
}

int main()
{
    int x1,x1,s1;
    x1 = sumx();
    x2 = sumx();
    s1 = x1 + x2;
    cout << "s1 = " << s1;

    int s2 = sumx() + sumx();
    cout << "s2 = " << sum;

    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

void sumx(int &x) {
    x = 10;
}

int main()
{
    int x1,x1,s1;
    sumx(x1);
    sumx(x2);
    s1 = x1 + x2;
    cout << "s1 = " << s1;

    //int s2 = sumx(x1) + sumx();
    //cout << "s2 = " << sum;

    return 0;
}
```

main.cpp

```
1  #include <iostream>
2  using namespace std;
3  void f1(int &x) {
4      x = 10;
5  }
6  void f2(int &x )
7  {
8      x = x+10;
9  }
10 int main()
11 {
12     int x1,x2,x3,x4;
13
14     f1(x1);
15     x2 = 0;
16     f1(x2);
17
18     f2(x3);
19     x4 = 20;
20     f2(x4);
21     return 0;
22 }
23
```

Problem?

Get single value

```cpp
#include <iostream>

using namespace std;

int getonedata() {
    int i;
    cout << " enter data : " ;
    cin>> i;
    return i;
}
int main()
{
    int x = getonedata();
    return 0;
}
```

```cpp

#include <iostream>

using namespace std;

void getonedata(int &i) {
    cout << " enter data : " ;
    cin>> i;
}
int main()
{
    int x;
    getonedata(x);
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

void getonedata(int &i) {
    cout << " enter data : " ;
    cin>> i;
}
int main()
{
    int x=30;
    getonedata(x);
    return 0;
}
```

Apirak :

Pass value and get value

```cpp
#include <iostream>

using namespace std;

void dosomething2(int &i) {
    i = i*3;
}
int main()
{
    int x=30;
    dosomething2(x);
    cout << x;
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

void dosomething1(int x,int &i) {
    i = x*3;

}
int main()
{
    int x=30;
    dosomething1(x, x);
    cout << x;
    return 0;
}
```

Ариак :

Return multiple value using pass by reference (1)

```cpp
#include <iostream>

using namespace std;

void dosomething3(int &i,int &j ,int &k) {
    cin >> i;
    cin >> j;
    cin >> k;
}
int main()
{
    int a,b,c;
    dosomething3(a,b,c);
    return 0;
}

```

main.cpp

```cpp
#include <iostream>

using namespace std;

int dosomething4(int &i,int &j) {
    cin >> i;
    cin >> j;
    cin >> k;
    return k;
}
int main()
{
    int a,b,c;
    c = dosomething4(a,b);
    return 0;
}

```

Return multiple value

using pass by reference (2)

```cpp
#include <iostream>

using namespace std;

void dosomething3(int &i,int &j ,int &k) {
    cin >> i;
    cin >> j;
    cin >> k;
}
int main()
{
    int a,b,c;
    dosomething3(a,b,c);
    return 0;
}

```

```cpp
#include <iostream>

using namespace std;

int dosomething5(int &i,int &j,int &k) {
    cin >> i;
    cin >> j;
    cin >> k;
    return 0;
}
int main()
{
    int a,b,c;
    if(dosomething5(a,b,c)==0)
      cout << "sucess";
    else
      cout << "fail";
    return 0;
}

```

Return status of function

Passing Array 2 Dimension in C/C++

main.cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

void GetData1 ( int Z [3 ] [3] )
{
   int i , j ;
   cout << "GetData1 ( int Z [3 ] [3] ) \n";
    for (  i = 0 ; i < 3 ; i++ )
    {
       for ( j = 0 ; j < 3 ; j++ )
       {
            Z[ i ] [ j ] = rand()%10;
            cout << setw(5) << Z[ i ] [ j ]  << "  ";
        }
      cout <<  endl;
    }
}

void GetData2 ( int Z [ ] [3] ,int numrow)
{
   int i , j ;

   cout << " ( int Z [ ] [3] ,int numrow)" << numrow <<"\n";
    for (  i = 0 ; i < numrow ; i++ )
    {
       for ( j = 0 ; j < 3 ; j++ )
       {
            Z[ i ] [ j ] = rand()%10;
            cout << setw(5) << Z[ i ] [ j ]  << "  ";
        }
      cout <<  endl;
    }
}
```

```cpp
/*
void GetData3 ( int Z [3 ] [] )
{
   int i , j ;

    for (  i = 0 ; i < 3 ; i++ )
    {
       for ( j = 0 ; j < 3 ; j++ )
       {
            Z[ i ] [ j ] = rand()%10;
            cout << setw(5) << Z[ i ] [ j ]  << "  ";
        }
      cout <<  endl;
    }
}
*/

int main()
{
    int A[3][3];
    int B[5][3];
    GetData1(A);
    GetData2(B,5);
    return 0;
}
```

Apirak :

ตัวอย่าง แบบ top down

จงเขียนโปรแกรมรับข้อมูลลงอาเรย์ A , B ขนาด 20x20 และหาผลคูณของอาเรย์ C โดย

$$C = A \times B$$

สูตรการคูณ คือ $C_{ij} = \sum a_{ik} b_{kj}$

เมื่อ $1 <= i <= 20$ และ $1 <= j <= 20$

Break down the problem

1. Get data both A and B
2. Calculate C = A*B;
3. Show C

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

void GetData ( int Z [3 ] [3] )
{
   int i , j ;

    for (  i = 0 ; i < 3 ; i++ )
    {
       for ( j = 0 ; j < 3 ; j++ )
       {
           //cout << " row = " << i << "column " << j ;
           //cin >> Z[ i ] [ j ] ;
           // Get a random number
            Z[ i ] [ j ] = rand()%10;
            cout << setw(5) << Z[ i ] [ j ]  << "  ";
        }
      cout <<  endl;
    }
}

void MultplyMatrix ( int W[ 3 ] [3 ],
                     int X[3][3],int Z[3][3 ] )
{
    int i , j , k ,sum ;
   for (  i = 0 ; i < 3 ; i++ )
    {
        for ( j = 0 ; j < 3 ; j++ )
        {
            sum = 0 ;
            for ( k = 0 ; k < 3 ; k++ )
            {
                sum  =sum + W[ i ] [ k ] * X [ k ] [ j ] ;
            }
            Z[ i ] [ j ] = sum ;
        }
   }
}
```

```
void PrintMatrix( int Z [3 ] [3] )
{
   int i , j ;
    cout << "value of  Matrix C \n" ;
   for (  i = 0 ; i < 3 ; i++ )
   {
      for ( j = 0 ; j < 3 ; j++ )
      {
            cout << setw(5) << Z[ i ] [ j ]  << "  ";
      }
      cout <<  endl;
   }
}
int main( )
{
    int  A[ 3 ] [ 3 ] ;
    int  B[ 3 ] [3] ;
    int  C[ 3 ] [ 3 ] ;
    int  i , j   ;

   cout << "input DATA of  Matrix A \n" ;
   GetData ( A ) ;
   cout << "input DATA of  Matrix B \n" ;
   GetData ( B ) ;
   MultplyMatrix ( A, B , C ) ;
   PrintMatrix(C);
}

```

Apirak :

main.cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

 void GetData ( int Z [ ] [3] )
{
   int i , j ;

    for (  i = 0 ; i < 3 ; i++ )
    {
       for ( j = 0 ; j < 3 ; j++ )
       {
            //cout << " row = " << i << "column " << j ;
            //cin >> Z[ i ] [ j ] ;
            // Get a random number
             Z[ i ] [ j ] = rand()%10;
             cout << setw(5) << Z[ i ] [ j ]  << "  ";
        }
      cout <<  endl;
    }
}
```

```cpp
void MultplyMatrix ( int W[  ] [3 ],
                     int X[ ][3],int Z[][3 ] )
{
    int i , j , k ,sum ;
   for (  i = 0 ; i < 3 ; i++ )
    {
        for ( j = 0 ; j < 3 ; j++ )
        {
            sum = 0 ;
             for ( k = 0 ; k < 3 ; k++ )
             {
                 sum  =sum + W[ i ] [ k ] * X [ k ] [ j ] ;
             }
            Z[ i ] [ j ] = sum ;
        }
   }
}
```

Apirak :

```
40
41  void PrintMatrix( int Z [ ] [3] )
42  {
43     int i , j ;
44      cout << "value of  Matrix C \n" ;
45     for (  i = 0 ; i < 3 ; i++ )
46     {
47        for ( j = 0 ; j < 3 ; j++ )
48        {
49              cout << setw(5) << Z[ i ] [ j ]  << "  ";
50        }
51        cout <<  endl;
52     }
53  }
54  int main( )
55  {
56      int  A[ 3 ] [ 3 ] ;
57      int  B[ 3 ] [3] ;
58      int  C[ 3 ] [ 3 ] ;
59      int  i , j   ;
60
61     cout << "input DATA of  Matrix A \n" ;
62     GetData ( A ) ;
63     cout << "input DATA of  Matrix B \n" ;
64     GetData ( B ) ;
65     MultplyMatrix ( A, B , C ) ;
66     PrintMatrix(C);
67  }
68
```

Apirak :

ตัวอย่าง (3)

จงเขียนโปรแกรมอ่านข้อมูล 2 ชนิด คือ A และ B ซึ่งชนิดละ20 จำนวน จากนั้นให้คำนวณค่า

A_bar = Sum i = 1 .. 20 ของ A

B_bar = Sum i = 1 .. 20 ของ B

$$Corr(A, B) = \frac{n*Sum(AiBi) - Sum(Ai)*Sum(Bi)}{sqrt\left(n*(sum(Ai*Ai) - Sum(Ai)*Sum(Ai)) * n*(sum(Bi*Bi) - Sum(Bi)*Sum(Bi))\right)}$$

กำหนดให้ z = sqrt(x) เป็นbuild in function ของการหาค่า root x

```
4
5  void GetData ( float Z[  ] ,int n );
6  float  Sum ( float Z[ ] , int n ) ;
7  void  MultAry ( float W[ ] , float X[ ] , float Z[ ] , int n ) ;
8
9
10 int main( )
11 {
12     float  A [ 20 ] , B[ 20 ] ,T[20] ;
13     float  A_bar , B_bar ,corrAB , X , Y;
14     float  AB_bar , AA_bar , BB_bar ;
15    cout << " input DATA of  Matrix A \n"  ;
16    GetData ( A ,20 ) ;
17    cout << " input DATA of  Matrix B \n"  ;
18    GetData ( B , 20  ) ;
19     A_bar = Sum ( A  , 20 ) ;
20     B_bar   = Sum ( B , 20 );
21     MultAry( A , A , T , 20 )  ; AA_bar = Sum ( T, 20 ) ;
22     MultAry(  B , B , T , 20 )  ; BB_bar = Sum ( T, 20 ) ;
23     MultAry( A , B , T , 20 )  ; AB_bar = Sum ( T, 20 ) ;
24
25     X   =  (20*AB_bar - A_bar*B_bar) ;
26     Y   = (  ( 20* (AA_bar - A_bar* A_bar) ) *
27                ( 20* (BB_bar - B_bar* B_bar) ) );
28     corrAB = X / sqrt ( Y ) ;
29
30     cout << "\ncorr(A,B)"  <<  corrAB ;
31
32     return 0;
33 }
34
```

Apirak :

https://www.w3resource.com/cpp-exercises

END

Apirak :

One dimensional array

ตัวอย่าง 3

```
//C++ method of passing Array to function is
//passing by reference
void GetData( float A [ ] , int n  ) ;
{
    int  i ;
    for ( i = 0 ; i < n ; i++)
    {
       cin >> A[ i ] ;
    }
}
```

One dimensional array

ตัวอย่าง 3

```
//C++ method of passing Array to function is
//passing by reference
float Sum( float X [ ] , int n  ) ;
{
    int  i  ;
    float s = 0 ;
     for ( i = 0 ; i < n ; i++)
     {
       s = s + X [ i ] ;
     }
     return s ;
}
```

```
void  MultAry ( float W[ ] , float X[ ] , float Z[ ] , int n ) ;
{
      int  i  ;
       for ( i = 0 ; i  < n ; i++)
       {
         Z[ i ]  = W [ i ]  * X [ i ] ;
       }
       return  ;
}
```

Passing One dimensional Array to function

One dimensional array

ตัวอย่าง 3

จงเขียนโปรแกรมหาค่าสูงสุดและต่ำสุด และตำแหน่งของค่าสูงและต่ำสุดของอาเรย์ Z ขนาด 120 เข้าสู่อาเรย์ Z

One dimensional array

ตัวอย่าง 3

```
void GetData( int A [120] ) ;
void main( )
{
    int z[120] , i ;
    int max , min ;
    int imax , imin ;

    GetData ( Z ) ;

    max = z[0] ; min = z[0];
    imax = 0;     imin = 0;
    for ( i = 1 ; i < 120 ; i++)
    {
       if (z[ i ] > max ) {
            max = z[ i ];
            imax = i ;

       }else if ( z[i] < min ) {
            min = z[ i ];
            imin = i ;
        }
    }
  cout << “max = ” << max ;
   cout << “min = ” << min ;
}
```

One dimensional array

ตัวอย่าง 3

```
//C++ method of passing Array to function is
//passing by reference
void GetData( int A [120] ) ;
{
    int  i  ;
     for ( i = 0 ; i < 120 ; i++)
     {
        cin >>A[ i ] ;
     }
}
```

One dimensional array

ตัวอย่าง 4

```
//C/C++ does not allow to return array data
//so this function is compile time error.
int  GetData( int A [120] ) ;
{
    int  i  ;
    for ( i = 0 ; i < 120 ; i++)
    {
       cin >>A[ i ] ;
    }
    return A
}
```

One dimensional array

ตัวอย่าง 3

```
void GetData(int A[ ] ,int n );
void main( )
{
    int z[120] y[10], i ;
    int max , min ;
    int imax , imin ;

     GetData ( Z , 120  ) ;
    GetData ( y , 10  ) ;

    max = z[0] ; min = z[0];
    imax = 0;     imin = 0;
```

```
for ( i = 1 ; i < 120 ; i++)
{
    if (z[ i ] > max ) {
          max = z[ i ];
          imax = i ;

   }else if ( z[i] < min ) {
          min = z[ i ];
          imin = i ;
     }
}
cout << “max = ” << max ;
 cout << “min = ” << min ;
}
```

One dimensional array

ตัวอย่าง 3

```
//C++ method of passing Array to function is
//passing by reference
void GetData( int A [ ] , int n  ) ;
{
    int  i ;
    for ( i = 0 ; i < n ; i++)
    {
       cin >> A[ i ] ;
    }
}
```

One dimensional array

ตัวอย่าง 3

จงเขียนโปรแกรมรับข้อมูลจำนวนเต็ม Z ขนาด 120 ตัว และหาค่าสูงสุดและต่ำสุด รวมทั้งหาตำแหน่งของค่าสูงและต่ำสุด

One dimensional array

ตัวอย่าง 3 (cont)

```
void main( )
{
   int z[120] , i ;
   int max , min ;
   int imax , imin ;
    for ( i = 0 ; i < 120 ; i++)
   {
      cin >> z[ i ] ;
    }
   max = z[0] ; min = z[0];
   imax = 0;     imin = 0;
   for ( i = 1 ; i < 120 ; i++)
   {
      if (z[ i ] > max ) {
            max = z[ i ];
            imax = i ;

      }else if ( z[i] < min ) {
            min = z[ i ];
            imin = i ;
       }
   }
  cout << “max = ” << max ;
   cout << “min = ” << min ;
}
```

One dimensional array

ตัวอย่าง 4 (cont)

```
void main( )
{
    int z[120] , i ;
    int max , min ;
    int imax , imin ;
     for ( i = 0 ; i < 120 ; i++)
    {
       cin >> z[ i ] ;
     }
    max = z[0] ; min = z[0];
    imax = 0;     imin = 0;
    for ( i = 1 ; i < 120 ; i++)
    {
       if (z[ i ] > max ) {
             max = z[ i ];
             imax = i ;

       }
        if ( z[i] < min ) {
             min = z[ i ];
             imin = i ;
         }
    }
   cout << “max = ” << max ;
    cout << “min = ” << min ;
}
```

```
#include<iostream.h>

float celsius_to_fahrenheit(float celsius);

int main()
 {
   float fahrenheit;
   float celsius = 22.5;

   fahrenheit = celsius_to_fahrenheit(celsius);
   cout << celsius << " C = " << fahrenheit << " F\n";

   return 0;
 }

float celsius_to_fahrenheit(float celsius)
 {
     float fahr = ( celsius * (9.0/5.0) + 32.0 );
     return  fahr ;
 }
```

```
C:\My Documents\sc107\files>ct
22.5 C = 72.5 F
```

```
#include<iostream.h>

float celsius_to_fahrenheit( float celsius );

int main()
 {
   float fahrenheit;
   float celsius = 22.5;

   fahrenheit = celsius_to_fahrenheit( celsius );
   cout << celsius << " C = " << fahrenheit << " F\n";

   return 0;
 }

float celsius_to_fahrenheit( float celsius )
{
    return ( celsius * (9.0/5.0) + 32.0 );
}
```

```
C:\My Documents\sc107\files>ct
22.5 C = 72.5 F
```

```
#include<iostream.h>
int celsius_to_fahrenheit( float celsius );
int main()
 {
  float fahrenheit;
  float celsius = 22.5;

  fahrenheit = celsius_to_fahrenheit( celsius );

  cout << celsius << " C = " << fahrenheit << " F\n";

  return 0;
 }

int celsius_to_fahrenheit( float celsius )
 {
    return (celsius * (9.0/5.0) + 32.0 );
 }
```

```
C:\My Documents\sc107\
22.5 C = 72 F
```

```
#include<iostream.h>
celsius_to_fahrenheit(float celsius);
int main()
 {
  float fahrenheit;
  float celsius = 22.5;
  fahrenheit = celsius_to_fahrenheit(celsius);
  cout << celsius << " C = " << fahrenheit << " F\n";
  return 0;
 }

celsius_to_fahrenheit(float celsius)
 {
  return(celsius * (9.0/5.0) + 32.0);
 }
```

```
C:\My Documents\sc107\file
22.5 C = 72 F
```

```
#include<iostream.h>
float celsius_to_fahrenheit(float celsius);
int main()
 {
  int fahrenheit ;
  float celsius = 22.5;
  fahrenheit = celsius_to_fahrenheit(celsius);
  cout << celsius << " C = " << fahrenheit << " F\n";
  return 0;
 }
float celsius_to_fahrenheit(float celsius)
{
  return(celsius * (9.0/5.0) + 32.0);
}
```

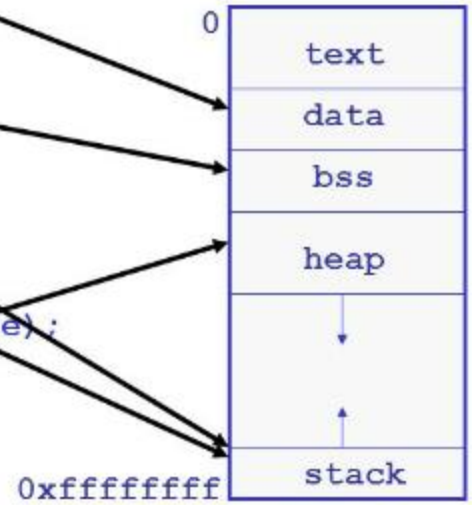
```
C:\My Documents\sc107\file
22.5 C = 72 F
```

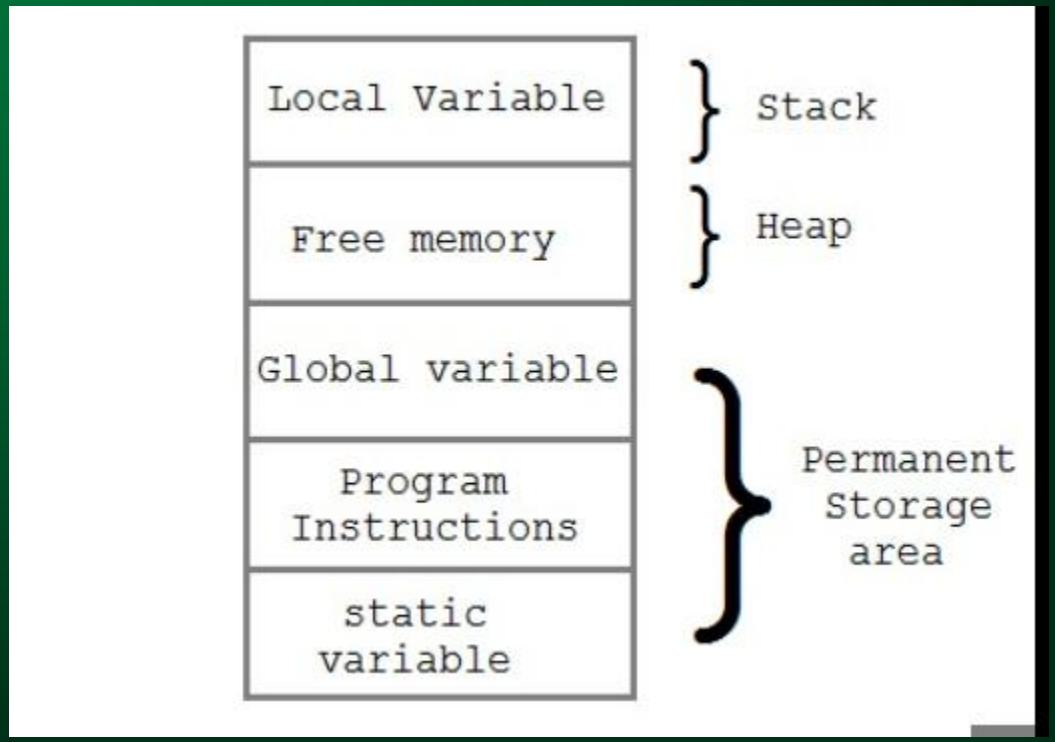
Recommend about "return"

- The return statement does not require that the valu begin returned be placed in parentheses.
- A function can return only one value using return statement
- When a return statement is encountered,the function will exit and return the value specified.

Example

```
char *string = "hello";

int iSize;

char *f (int x)
{
  char *p;

  iSize = 8;
  p = (char *)malloc (iSize);

  return p;
}
```




Local Variable
Free memory
Global variable
Program Instructions
static variable
Stack
Heap
Permanent Storage area

C/C++ Structure Type & Function

- C/C++ support both pass by value & pass by reference

Array with Function in C/C++

- Primitive type : int , float , long , double , char
- User define type : structure
- Array is using “pass by reference”.

Pass array to function

```
8
9   #include <iostream>
10  using namespace std;
11  void updateArray( int data[ ],int n )
12  {
13      for(int i = 0 ; i < n ; i++) {
14          data[i] = data[i]+1000;
15      }
16  }
17  void showarray( int data[ ],int n )
18  {
19      for(int i = 0 ; i < n ; i++) {
20          cout << data[i] << ",";
21      }
22  }
23  int main() {
24
25    int Roll_Number[5] = {100,101,102,103,104};
26
27      cout << "\nbefore update \n";
28      showarray(Roll_Number,5);
29      updateArray(Roll_Number,3);
30      cout << "\nafter update \n";
31      showarray(Roll_Number,5);
32  }
```

```
before update
100,101,102,103,104,
after update
1100,1101,1102,103,104,
```

Return data with array in C/C++

- C/C++ not allow the code return data in array form

Run | Debug | Stop | Share

main.cpp

```cpp
#include <iostream>

using namespace std;

int * dosomething6() {
    int x[5]={0};

    for(int i=0 ; i < 5 ;i++)
        x[i] = i;
    return x;
}

int main()
{
    int x[5] ;
    x = dosomething6();
      for(int i=0 ; i < 5 ;i++)
        cout << x[i] << " ";
    return 0;
}
```

input

Compilation failed due to following error(s).

```
main.cpp: In function ‘int* dosomething6()’:
main.cpp:10:12: warning: address of local variable ‘x’ returned [-Wreturn-local-addr
   10 |     return x;
      |            ^
main.cpp:6:9: note: declared here
    6 |     int x[5]={0};
      |         ^
main.cpp: In function ‘int main()’:
main.cpp:16:7: error: incompatible types in assignment of ‘int*’ to ‘int [5]’
   16 |     x = dosomething6();
      |     ~~^~~~~~~~~~~~~~~~
```

Getting value in array using passing parameter

main.cpp

```cpp
//
#include <iostream>

using namespace std;

void dosomething7(int x[],int n) {
    for(int i=0 ; i < n ;i++)
        x[i] = i;
}
int main()
{
    int x[5] = {0};
    dosomething7(x,5);
      for(int i=0 ; i < 5 ;i++)
        cout << x[i] << " ";
    return 0;
}

```

main.cpp

```cpp
#include <iostream>
using namespace std;

struct employee {
 char name[100];
 int age;
 float salary;
 char department[50];
};

// This function takes structure variable as parameter
void printEmployeeDetails(struct employee emp){
   cout << "\n*** Employee Details ***\n";
   cout << "Name : " << emp.name << "\nAge : "<< emp.age << "\nSalary : "
        << emp.salary << "\nDepartment : "<< emp.department;
}
int main(){
   struct employee manager ;

   printf("Enter Name, Age, Salary and Department of Employee\n");
   // Assigning data to members of structure variable
   cin >> manager.name >> manager.age >> manager.salary >> manager.department;

   // Passing structure variable to function
   printEmployeeDetails(manager);
}
```

```
Enter Name, Age, Salary and Department of Employee
apirak
30
1000.5
ram

*** Employee Details ***
Name : apirak
Age : 30
Salary : 1000.5
Department : ram
```

```
8
9   #include <iostream>
10  using namespace std;
11  struct myStructure {
12      int x;
13      int y;
14  };
15
16  void swap(myStructure &ms)
17  {
18      int temp = ms.x;
19      ms.x = ms.y;
20      ms.y = temp;
21  }
22
23  int main()
24  {
25      myStructure ms = { 50, 60 };
26
27      cout<<"\n\nBefore swap, inside main():\n";
28      cout<<"x = "<<ms.x<<"\ty = "<<ms.y;
29
30      swap(ms);
31
32      cout<<"\n\nAfter swap, inside main():\n";
33      cout<<"x = "<<ms.x<<"\ty = "<<ms.y;
34      cout<<endl;
35      return 0;
36  }
```

```
Before swap, inside main():
x = 50   y = 60

After swap, inside main():
x = 60   y = 50
```

Function return Structure Type in C/C++

```
4  struct product {
5     char id[10];
6     char name[30];
7     float cost;
8     int instock;
9  };
10
11 struct product  getNewProduct() {
12   struct product p;
13   cout << "New product: " << endl;
14   cout << "id =? ";  cin >> p.id;
15   cout << "name=? "; cin >> p.name;
16   cout << "cost=? "; cin >> p.cost;
17   cout << "instock=? "; cin >> p.instock;
18   return p;
19 }
20 int main() {
21    struct product newProduct = getNewProduct();
22    cout << "\nReturn product: " << endl;
23    cout << "\nid =? " <<  newProduct.id;
24    cout << "\nname=? " << newProduct.name;
25    cout << "\ncost=? " << newProduct.cost;
26    cout << "\ninstock=? "<<  newProduct.instock;
27 }
28
```

```
New product:
id =? 10
name=? api
cost=? 50.3
instock=? 50

Return product:

id =? 10
name=? api
cost=? 50.3
instock=? 50
```

Menu

1. Add item to Cart
2. Sell Product
3. Check Stock
4. End Program

Please select 1 / 2 / 3/ 4 =?

Ex 2 จงใช้โครงสร้างข้อมูลแบบ record

เงินเดือนของพนักงาน XYZ ขึ้นอยู่กับ ฐานเงินเดือนที่เท่ากันทุกคน คือ 3000 บาท บวกกับเปอร์เซนต์ของฐานเงินเดือนตาม ชนิดของงาน จำนวนปีที่ทำงาน ระดับความรู้ ความเอาใจใส่งาน

รายได้ (ic) = เงินเดือน + ฐานเงินเดือน * (%

ชนิดของงาน	%
1	0
2	5
3	15

ระดับความรู้	%
1 (ป.ว.ช)	0
2 (ป.ว.ส)	5
3 (ป.ตรี)	15

Ex 2 จงใช้โครงสร้างข้อมูลแบบ record (ต่อ)

จงเขียนโปรแกรมรับตัวเลขรหัสของ 4 รายการข้างต้นของพนักงานแต่ละคน เพื่อคำนวณเงินเดือน จำนวน 120 คน

จำนวนปีที่ทำงาน	%
0 - 10	5
แต่ละปีถัดไป	4

ความเอาใจใส่งาน	%
0 (ไม่ดี)	0
1 (ดี)	10
2 (ดีมาก)	25

Ex 2 จงใช้โครงสร้างข้อมูลแบบ record

เงินเดือนของพนักงาน XYZ ขึ้นอยู่กับ ฐานเงินเดือนที่เท่ากันทุกคน คือ 3000 บาท บวกกับเปอร์เซนต์ของฐานเงินเดือนตาม ชนิดของงาน จำนวนปีที่ทำงาน ระดับความรู้ ความเอาใจใส่งาน

รายได้ (ic) = เงินเดือน + ฐานเงินเดือน * (% ชนิดงาน)

+ ฐานเงินเดือน * (%จำนวนปีที่ทำงาน **)**

+ ฐานเงินเดือน * (%ความเอาใจใส่งาน **)**

แบบฝึกหัด 2

จงเขียนโปรแกรมต้องการนำระเบียนลูกค้า(customer) ของสหกรณ์ออมทรัพย์รามคำแหง ที่รองรับการจัดเก็บข้อมูลไม่เกิน 500 คน ระเบียนลูกค้าประกอบด้วย รหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก(Deposit)

Menu

1. Open a new Account
2. Deposit
3. Withdraw
4. Compound interest
5. Quit

Please select 1 / 2 / 3 / 4 / 5 =?

เลือก 1 เป็นการเปิดบัญชีใหม่

ระบบจะแสดงหน้าจอให้ผู้ใช้ ป้อนรหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก (Deposit) โดยรหัสบัญชีต้องไม่ซ้ำกัน

เลือก 2 เป็นการฝากเงิน

โดยผู้ใช้ป้อนรหัสบัญชี(id)ที่ต้องการ ระบบจะตรวจสอบเพื่อค้นหาระเบียนลูกค้า เพื่อให้ผู้ใช้งานป้อนยอดฝาก เพื่อปรับปรุงบัญชีให้ถูกต้อง

เลือก 3 เป็นการถอนเงิน

โดยผู้ใช้ป้อนรหัสบัญชี(id)ที่ต้องการ ระบบจะตรวจสอบเพื่อค้นหาระเบียนลูกค้า เพื่อให้ผู้ใช้งานป้อนยอดถอนแต่ต้องมีเงินติดบัญชีอย่างน้อย 100 บาท ระบบทำการปรับปรุงบัญชีให้ถูกต้อง

เลือก 4 เป็นการคิดดอกเบี้ยทบต้น

โดยระบบจะให้ผู้ใช้ป้อนอัตราดอกเบี้ยต่อปี(rate) ทางแป้นพิมพ์ เพื่อคิดดอกเบี้ยทบต้นของทุกบัญชี ซึ่งกระทำทุกวันได้ดอกเบี้ยเป็นรายวัน โปรแกรมจะแสดงเงินฝากเก่า ดอกเบี้ยที่ได้รับ และเงินฝากใหม่(เงินฝากเดิม+ดอกเบี้ยที่ได้)ปรากฏบน

เงินฝากใหม่ = เงินฝากเดิม+ดอกเบี้ยที่ได้

จอภาพ : ดอกเบี้ยที่ได้ =อัตราดอกเบี้ยที่ได้*เงินฝากเดิม

แบบฝึกหัด 2

จงเขียนโปรแกรมต้องการนำระเบียนลูกค้า(customer) ของสหกรณ์ออมทรัพย์รามคำแหง เก็บในตัวแปรแถวลำดับ 1 มิติ ระเบียนลูกค้าประกอบด้วย รหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก(Deposit)

Menu

1. Open a new Account
2. Deposit
3. Withdraw
4. Compound interest
5. Quit

Please select 1 / 2 / 3 / 4 / 5 =?

<u>เลือก 1</u> เป็นการเปิดบัญชีใหม่

ระบบจะแสดงหน้าจอให้ผู้ใช้ ป้อนรหัสบัญชี(id) ชื่อบัญชี(name) ที่อยู่(Address) และเงินฝาก (Deposit) โดยรหัสบัญชีต้องไม่ซ้ำกัน

<u>เลือก 2</u> เป็นการฝากเงิน

โดยผู้ใช้ป้อนรหัสบัญชี(id)ที่ต้องการ ระบบจะตรวจสอบเพื่อค้นหาระเบียนลูกค้า เพื่อให้ผู้ใช้งานป้อนยอดฝาก เพื่อปรับปรุงบัญชีให้ถูกต้อง

เลือก 3 เป็นการถอนเงิน

โดยผู้ใช้ป้อนรหัสบัญชี(id)ที่ต้องการ ระบบจะตรวจสอบเพื่อค้นหาระเบียนลูกค้า เพื่อให้ผู้ใช้งานป้อนยอดถอนแต่ต้องมีเงินติดบัญชีอย่างน้อย 100 บาท ระบบทำการปรับปรุงบัญชีให้ถูกต้อง

เลือก 4 เป็นการคิดดอกเบี้ยทบต้น

โดยระบบจะให้ผู้ใช้ป้อนอัตราดอกเบี้ยต่อปี(rate) ทางแป้นพิมพ์ เพื่อคิดดอกเบี้ยทบต้นของทุกบัญชี ซึ่งกระทำทุกวันได้ดอกเบี้ยเป็นรายวัน โปรแกรมจะแสดงเงินฝากเก่า ดอกเบี้ยที่ได้รับ และเงินฝากใหม่ปรากฏบนจอภาพ

เลือก 5 เป็นการจบการทำงานซ้ำ

Array 2 dimension Passing In C/C++

ตัวอย่าง แบบ top down

จงเขียนโปรแกรมรับข้อมูลลงอาเรย์ A , B ขนาด 20x20 และหาผลคูณของอาเรย์ C โดย

$$C = A \times B$$

สูตรการคูณ คือ $C_{ij} = \sum a_{ik} b_{kj}$

เมื่อ $1 <= i <= 20$ และ $1 <= j <= 20$

Break down the problem

1. Get data both A and B
2. Calculate C = A*B;
3. Show C

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

void GetData ( int Z [3 ] [3] )
{
   int i , j ;

    for (  i = 0 ; i < 3 ; i++ )
    {
       for ( j = 0 ; j < 3 ; j++ )
       {
           //cout << " row = " << i << "column " << j ;
           //cin >> Z[ i ] [ j ] ;
           // Get a random number
            Z[ i ] [ j ] = rand()%10;
            cout << setw(5) << Z[ i ] [ j ]  << "  ";
        }
      cout <<  endl;
    }
}

void MultplyMatrix ( int W[ 3 ] [3 ],
                     int X[3][3],int Z[3][3 ] )
{
    int i , j , k ,sum ;
   for (  i = 0 ; i < 3 ; i++ )
    {
        for ( j = 0 ; j < 3 ; j++ )
        {
            sum = 0 ;
            for ( k = 0 ; k < 3 ; k++ )
            {
                sum  =sum + W[ i ] [ k ] * X [ k ] [ j ] ;
            }
            Z[ i ] [ j ] = sum ;
        }
   }
}
```

```
41 void PrintMatrix( int Z [3 ] [3] )
42 {
43    int i , j ;
44     cout << "value of  Matrix C \n" ;
45    for (  i = 0 ; i < 3 ; i++ )
46    {
47       for ( j = 0 ; j < 3 ; j++ )
48       {
49             cout << setw(5) << Z[ i ] [ j ]  << "  ";
50       }
51       cout <<  endl;
52    }
53 }
54 int main( )
55 {
56     int  A[ 3 ] [ 3 ] ;
57     int  B[ 3 ] [3] ;
58     int  C[ 3 ] [ 3 ] ;
59     int  i , j   ;
60
61    cout << "input DATA of  Matrix A \n" ;
62    GetData ( A ) ;
63    cout << "input DATA of  Matrix B \n" ;
64    GetData ( B ) ;
65    MultplyMatrix ( A, B , C ) ;
66    PrintMatrix(C);
67 }
68
```

Apirak :

Apirak :

main.cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{
     int  A[ 4 ] [ 5 ] ;
     int  i , j ;
     int max , col_max , row_max ;

     srand((unsigned) time(NULL));
    cout << "input DATA of  Matrix A \n" ;
    for (  i = 0 ; i < 4 ; i++ )
    {
       for ( j = 0 ; j < 5 ; j++ )
       {
            //cout <<"row = " << i << "  column " << j << " " ;
            //cin >> A[ i ] [ j ] ;
             A[ i ] [ j ] = rand()%15;
             //cout << "A[" << i << "] [" << j << "] " <<  A[ i ] [ j ] << endl;
             cout << setw(5) << A[ i ] [ j ]  << "  ";
       }
       cout <<  endl;
    }

```

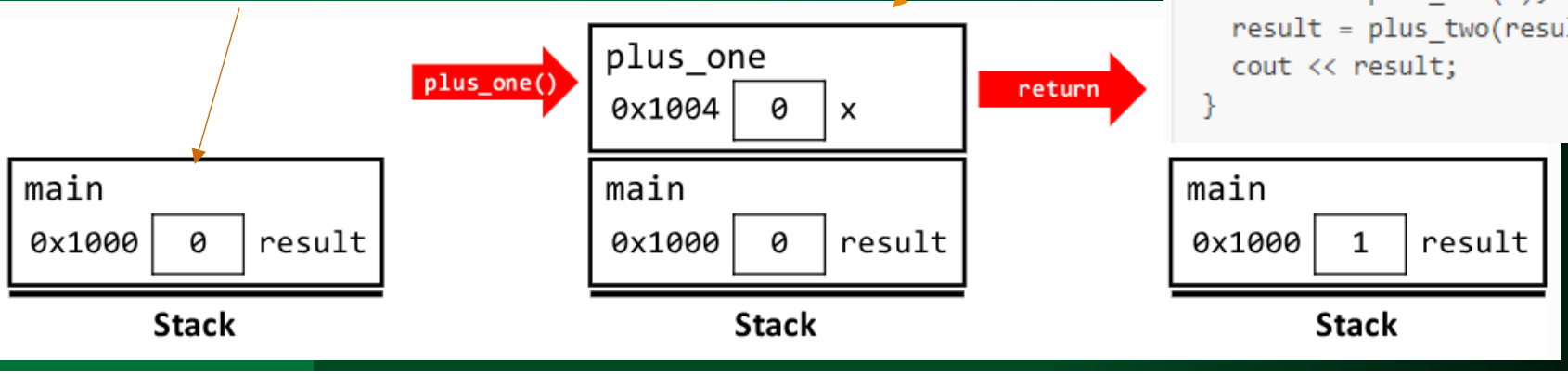
Output

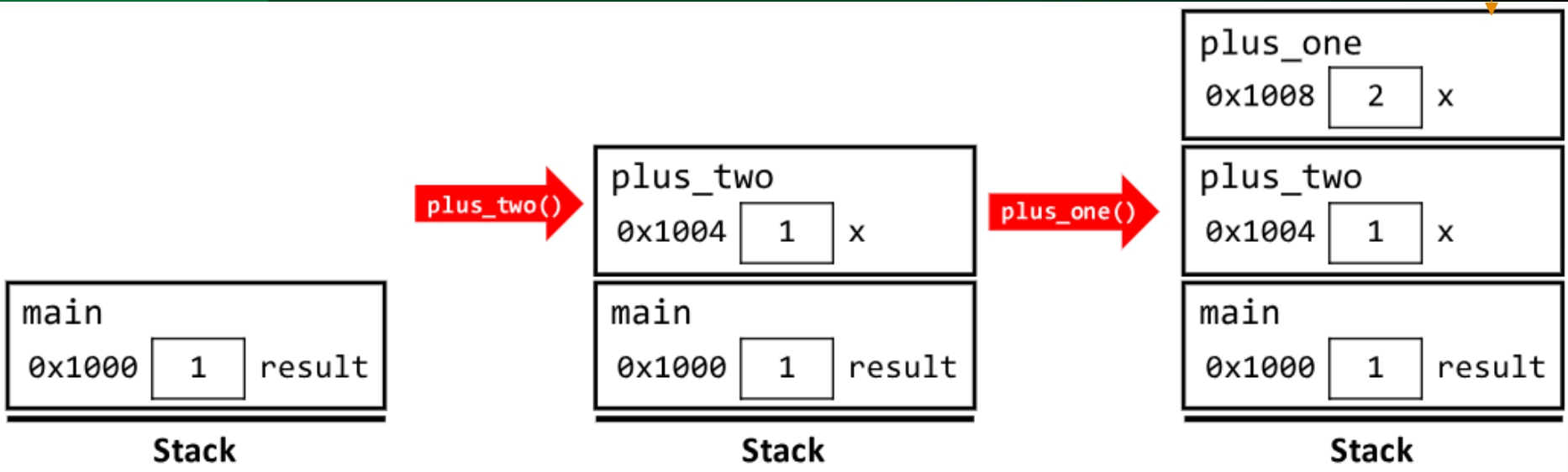
```
/tmp/CDh1DOGEgA.o
input DATA of  Matrix A
    7      8     11     10     11
    2      4      2      2      3
    1     14      8      9     14
    5      5     14      6      5
max value = 14
 in column = 1
 in row  =  2
```

Execution flows (4)

OS call

```
int plus_one(int x) {
  return x + 1;
}

int plus_two(int x) {
  return plus_one(x + 1);
}

int main() {
  int result = 0;
  result = plus_one(0);
  result = plus_two(result);
  cout << result;              // prin
}
```





return array (1)

main.cpp

```cpp
#include <iostream.h>

//using namespace std;

int *dosomething6() {
    int x[5]={0};
    for(int i=0 ; i < 5 ;i++)
        x[i] = i;
    return x;
}
int main()
{
    int *x;
    x = dosomething6();
      for(int i=0 ; i < 5 ;i++)
        cout << x[i] << " ";
    return 0;
}
```

main.cpp

```cpp
#include <iostream>

using namespace std;

int *dosomething6() {
    int x[5]={0};
    for(int i=0 ; i < 5 ;i++)
        x[i] = i;
    return x;
}
int main()
{
    int x[5] ;
    x = dosomething6();
      for(int i=0 ; i < 5 ;i++)
        cout << x[i] << " ";
    return 0;
}
```

Compilation failed due to following error(s).

```
main.cpp: In function ‘int* dosomething6()’:
main.cpp:9:12: warning: address of local variable ‘x’ returned [-Wreturn-local-addr]
    9 |     return x;
      |            ^
main.cpp:6:9: note: declared here
    6 |     int x[5]={0};
      |         ^
main.cpp: In function ‘int main()’:
main.cpp:14:7: error: incompatible types in assignment of ‘int*’ to ‘int [5]’
   14 |     x = dosomething6();
      |     ~~^~~~~~~~~~~~~~~~
```

Return local array

main.cpp

```cpp
#include <iostream>

using namespace std;

int *dosomething6() {
    int x[5]={0};
    for(int i=0 ; i < 5 ;i++)
        x[i] = i;
    return x;
}
int main()
{
    int x[5] ;
    x = dosomething6();
      for(int i=0 ; i < 5 ;i++)
        cout << x[i] << " ";
    return 0;
}
```

main.cpp

```cpp
#include <iostream>

using namespace std;

void dosomething7(int x[],int n) {
    for(int i=0 ; i < n ;i++)
        x[i] = i;
}
int main()
{
    int x[5] ;
    dosomething7(x,5);
      for(int i=0 ; i < 5 ;i++)
        cout << x[i] << " ";
    return 0;
}
```