Variable Arguments Handling in C.

[intro]

There is a sequence of function to handle variable arguments in C which is defined in the library -- <stdarg.h>.

What are variable arguments?

Consider the following function call.

If your own function want to retrieve many arguments (how many you don’t know before, it varies in runtime).

[namespace]

NONE

[library]

<stdarg.h>

[method 1]

The first method to achieve it is to use the array.

Define your own function with a parameter which type is an array.

Before calling your own function, pack it into array or other containers (which is defined in one of C++ STL).

Then call your own function.

Look at the following code.

In the following code, I use 3 statements to assign variable into vector, one by one.

In fact, you must assign it into array or other container one by one.

It’s too troublesome, isn’t it?

Hence, in the second method, I will teach you how to handle variable arguments list using va\_arg().

[P.S.]

(1)There is similar implementation in Python and MatLab (but it use a sequence of symbol) and

it is more convenient to use.

(2)When I wrote peterson.c (which is placed in GitHub),

this is the first time to know how to handle variable arguments list in C.

Before that time, I think there is NO way to handle variable arguments list.

[code]

#include <iostream>

#include <vector>

using namespace std;

void PrintArray(vector<int> vec1)

{

for(int i=0;i<vec1.size();i++){cout<<vec1[i]<<” “;}

cout<<endl;

}

int main()

{

int x,y,z;

vector<int> vec;

x=6;

y=7;

z=8;

vec.push\_back(x);

vec.push\_back(y);

vec.push\_back(z);

return 0;

}

[method 2]

Using the function va\_arg().

If you don’t what it means, I recommend you look at the following code first.

Step(1):

Define your own function which first parameter is int and unknown how many parameters.

Step(2):

Define a variable which type is va\_list using va\_list.

Step(3):

Call the function va\_start() in your own function to retrieve the other parameters which is defined in your own function.

Step(4):

Use the function va\_list() in your own function to access the element of variable arguments list.

Step(5):

After finishing, call the function va\_end().

It is abstract and is hard to understand, isn’t it?

Let’s look at the following code.

[code]

#include <stdio.h>

#include <stdarg.h>

void PrintFloats (int n, ...)

{

int i;

double val;

printf ("Printing floats:");

va\_list vl;

va\_start(vl,n);

for (i=0;i<n;i++)

{

val=va\_arg(vl,double);

printf (" [%.2f]",val);

}

va\_end(vl);

printf ("\n");

}

int main ()

{

PrintFloats (3,3.14159,2.71828,1.41421);

return 0;

}

[result]

Printing floats: [3.14] [2.72] [1.41]

In this example, in the function call PrintFloats(), the first argument – 3 refers that there are three variable arguments to handle.

In the definition of PrintFloats(), the first parameter named n means it is expected there are (n+1) in the function call.

After first parameter, it is represented by a “…”.

In the function of PrintFloats(), the function va\_start() is used for initialize the variable argument list.

In the function of PrintFloats(), the function va\_list() is used for retrieve variables from the argument list from top to back.(FIFO data structure)

In the function of PrintFloats(), the function va\_end() is used for telling handling of the variable argument list is done.

[ref]

<https://cplusplus.com/reference/cstdarg/va_arg/>