

Function (1A)

Copyright (c) 2010, 2011 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Task: Finding Partial Sums (1)

$$S_n = \sum_{k=1}^n a_k$$

$$a_k = k$$

$$S_1 = \sum_{k=1}^1 k = 1$$

```
printf("S_1 = %d \n", S_1);
```

$$S_2 = \sum_{k=1}^2 k = 1 + 2$$

```
printf("S_2 = %d \n", S_2);
```

$$S_3 = \sum_{k=1}^3 k = 1 + 2 + 3$$

```
printf("S_3 = %d \n", S_3);
```

Task: Finding Partial Sums (2)

$$S_1 = \sum_{k=1}^1 k = 1$$

$$S_2 = \sum_{k=1}^2 k = 1 + 2$$

$$S_3 = \sum_{k=1}^3 k = 1 + 2 + 3$$

```
S_1 = 0;  
for (k=1; k<=1; ++k) S_1 += k;
```

```
printf("S_1 = %d \n", S_1);
```

```
S_2 = 0;  
for (k=1; k<=2; ++k) S_2 += k;
```

```
printf("S_2 = %d \n", S_2);
```

```
S_3 = 0;  
for (k=1; k<=3; ++k) S_3 += k;
```

```
printf("S_3 = %d \n", S_3);
```

Task: Finding Partial Sums (3)

```
ni = 1;
{
  int n = ni;
  int S = 0;
  for (k=1; k<=n; ++k) S += k;
}
S_1 = S;
```

```
printf("S_1 = %d \n", S_1);
```

```
ni = 2;
{
  int n = ni;
  int S = 0;
  for (k=1; k<=n; ++k) S += k;
}
S_2 = S;
```

```
printf("S_2 = %d \n", S_2);
```

```
ni = 3;
{
  int n = ni;
  int S = 0;
  for (k=1; k<=n; ++k) S += k;
}
S_3 = S;
```

```
printf("S_3 = %d \n", S_3);
```

```
int psum(int n)
{
  int S = 0;
  for (k=1; k<=n; ++k) S += k;
  return S;
}
```

Task: Finding Partial Sums (4)

```
int psum(int n)
{
    int S = 0;
    for (k=1; k<=n; ++k) S += k;
    return S;
}
```

```
S_1 = psum ( 1 );
```

```
printf("S_1 = %d \n", S_1);
```

```
S_2 = psum ( 2 );
```

```
printf("S_1 = %d \n", S_2);
```

```
S_3 = psum ( 3 );
```

```
printf("S_1 = %d \n", S_3);
```

ni = 1;

S_1 = S;

ni = 2;

S_2 = S;

ni = 3;

S_1 = S;

```
ni = □;
```

```
{
    int n = ni;
    int S = 0;
    for (k=1; k<=n; ++k) S += k;
}
```

```
□ = S;
```

Task: Finding Partial Sums (5)

```
int psum(int n)
```

```
{  
    int S = 0;  
    for (k=1; k<=n; ++k) S += k;  
    return S;  
}
```

Function Definition

```
S_1 = psum ( 1 ); // Function Call
```

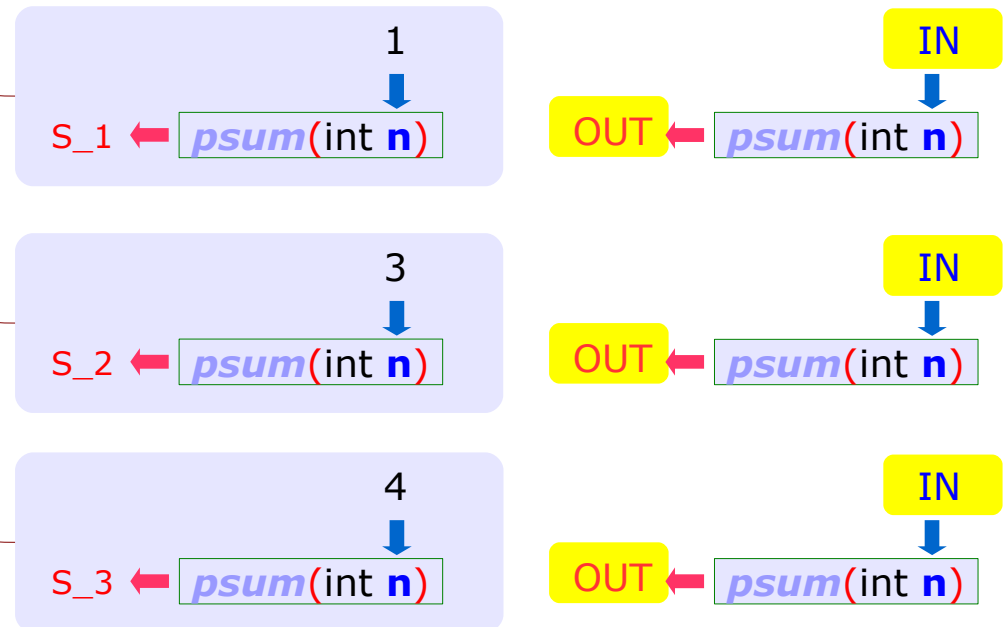
```
printf("S_1 = %d \n", S_1);
```

```
S_2 = psum ( 2 ); // Function Call
```

```
printf("S_1 = %d \n", S_2);
```

```
S_3 = psum ( 3 ); // Function Call
```

```
printf("S_1 = %d \n", S_3);
```



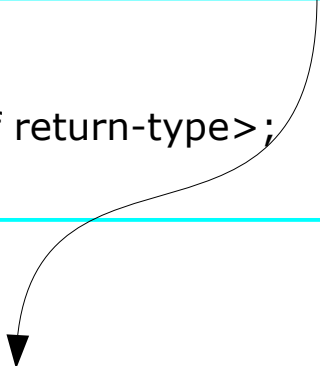
Function Definition

```
int psum(int n)
{
    int S = 0;
    for (k=1; k<=n; ++k) S += k;
    return S;
}
```

```
int mult(int m, int n)
{
    return m*n;
}
```

```
<return-type> function-name ( <parameter-list> )
{
    <statements>
    return <expression of return-type>;
}
```

(data-type var, data-type var, data-type var ,)



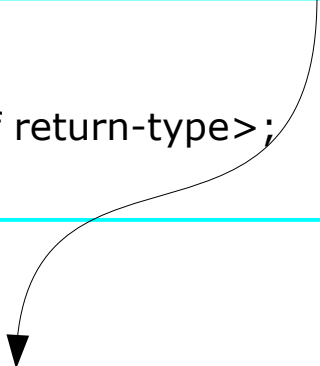
Function Declaration

```
int psum(int n)
{
    int S = 0;
    for (k=1; k<=n; ++k) S += k;
    return S;
}
```

```
int mult(int m, int n)
{
    return m*n;
}
```

```
<return-type> function-name ( <parameter-list> )
{
    <statements>
    return <expression of return-type>;
}
```

(data-type var, data-type var, data-type var ,)



2-d Array

References

- [1] Essential C, Nick Parlante
- [2] Efficient C Programming, Mark A. Weiss
- [3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
- [4] C Language Express, I. K. Chun