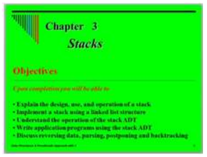


Chapter 3

Stacks

Objectives

Upon completion you will be able to:

- Explain the design, use, and operation of a stack
- Implement a stack using a linked list structure
- Understand the operation of the stack ADT
- Write application programs using the stack ADT
- Discuss reversing data, parsing, postponing and backtracking

CS 1037
Computer Science Fundamentals II

Part Nine:
Stacks

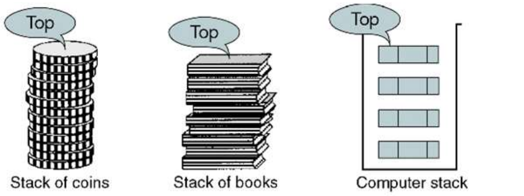


FIGURE 3-1 Stack

3-1 Basic Stack Operations

The stack concept is introduced and three basic stack operations are discussed.

- Push
- Pop
- Stack Top

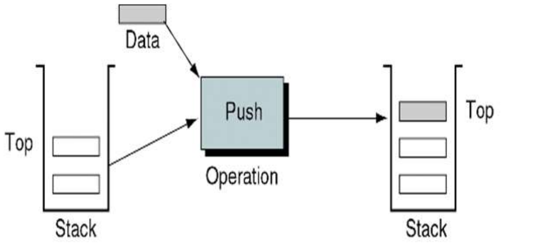


FIGURE 3-2 Push Stack Operation

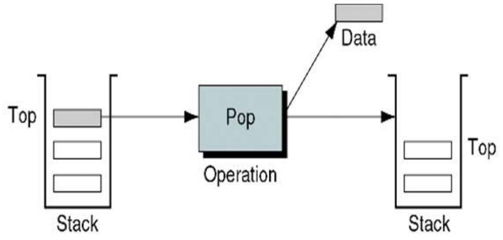


FIGURE 3-3 Pop Stack Operation

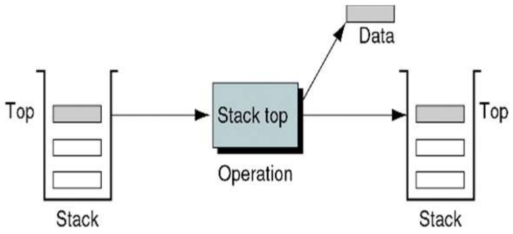


FIGURE 3-4 Stack Top Operation

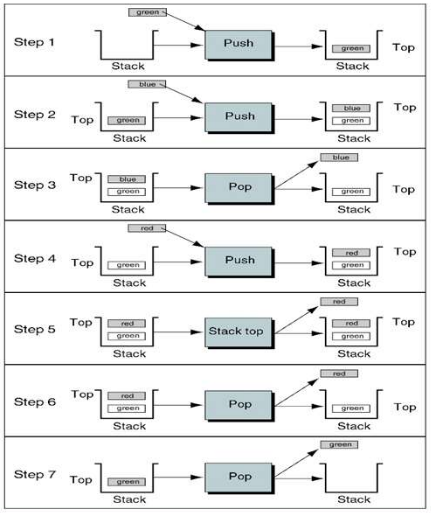
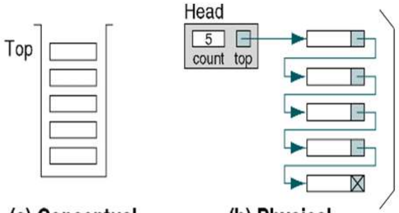
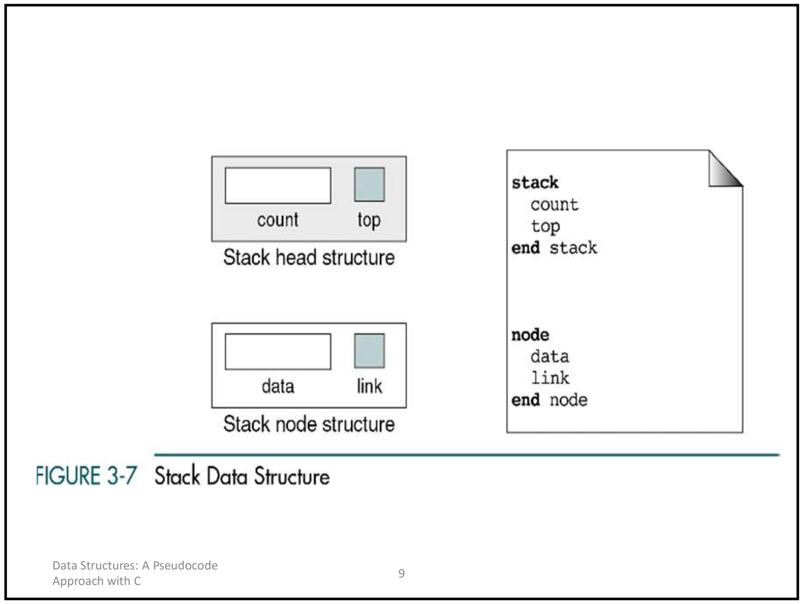


FIGURE 3-5 Stack Example



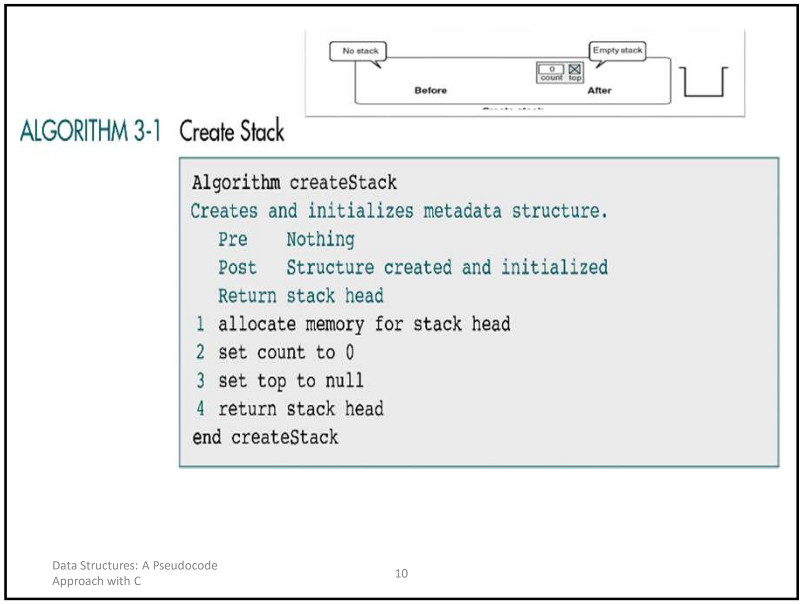
(a) Conceptual (b) Physical

FIGURE 3-6 Conceptual and Physical Stack Implementations



```
stack
  count
  top
end stack

node
  data
  link
end node
```

FIGURE 3-7 Stack Data Structure



No stack
Empty stack
0
count
top
Before
After

ALGORITHM 3-1 Create Stack

```
Algorithm createStack
Creates and initializes metadata structure.
   Pre    Nothing
   Post   Structure created and initialized
   Return stack head
   allocate memory for stack head
   set count to 0
   set top to null
   return stack head
end createStack
```

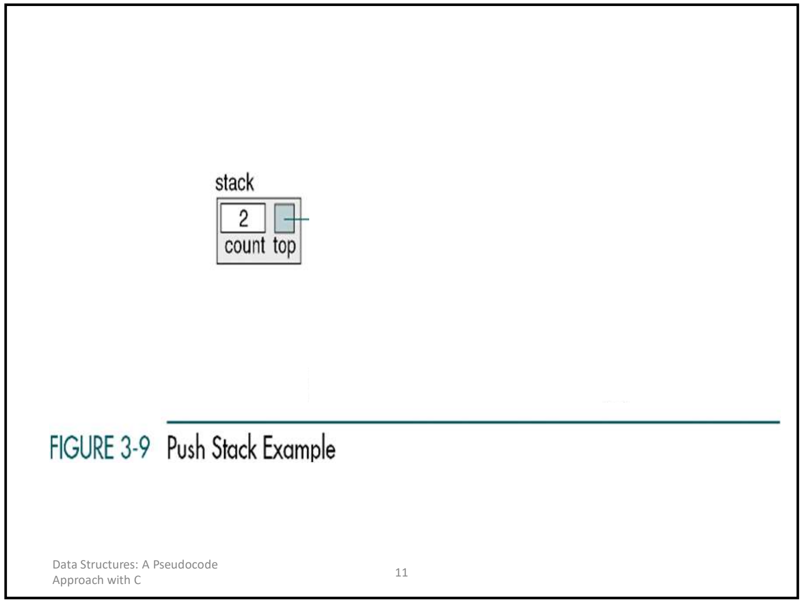
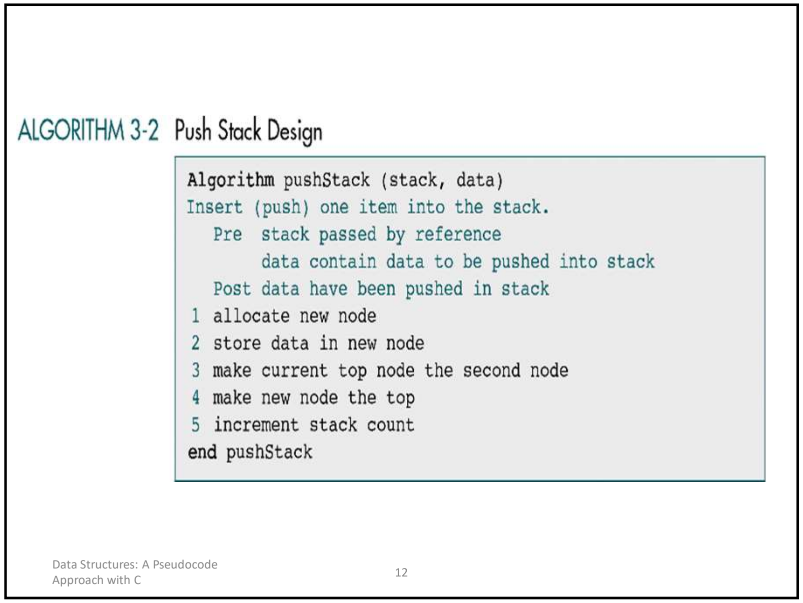


FIGURE 3-9 Push Stack Example



ALGORITHM 3-2 Push Stack Design

```
Algorithm pushStack (stack, data)
Insert (push) one item into the stack.
   Pre  stack passed by reference
        data contain data to be pushed into stack
   Post data have been pushed in stack
   allocate new node
   store data in new node
   make current top node the second node
   make new node the top
   increment stack count
end pushStack
```

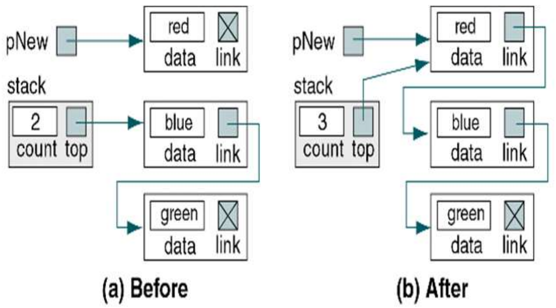
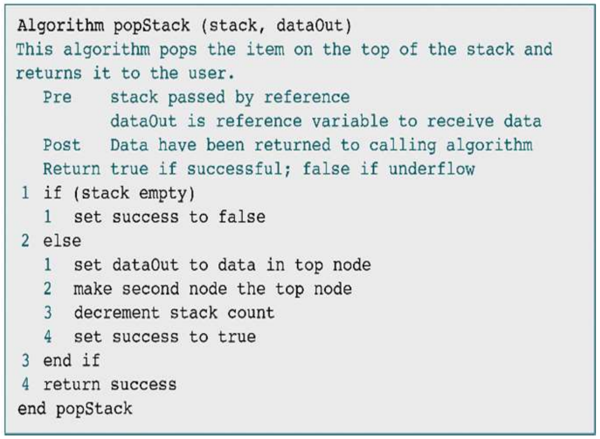


FIGURE 3-9 Push Stack Example

ALGORITHM 3-3 Pop Stack



```
Algorithm popStack (stack, dataOut)
This algorithm pops the item on the top of the stack and
returns it to the user.
   Pre    stack passed by reference
          dataOut is reference variable to receive data
   Post   Data have been returned to calling algorithm
   Return true if successful; false if underflow
1  if (stack empty)
   1  set success to false
2  else
   1  set dataOut to data in top node
   2  make second node the top node
   3  decrement stack count
   4  set success to true
3  end if
4  return success
end popStack
```

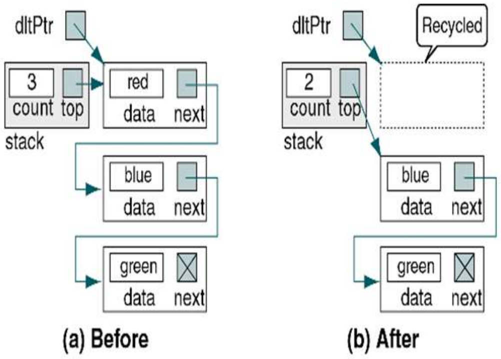
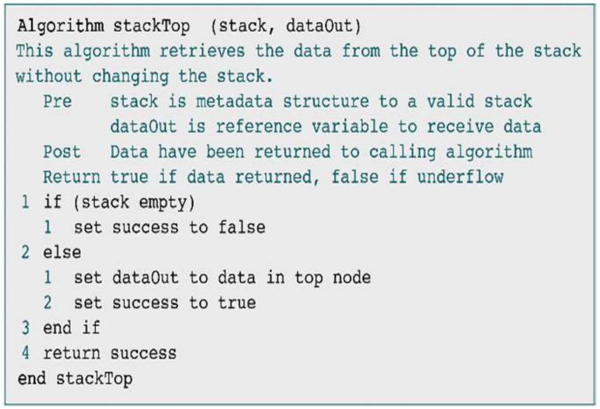
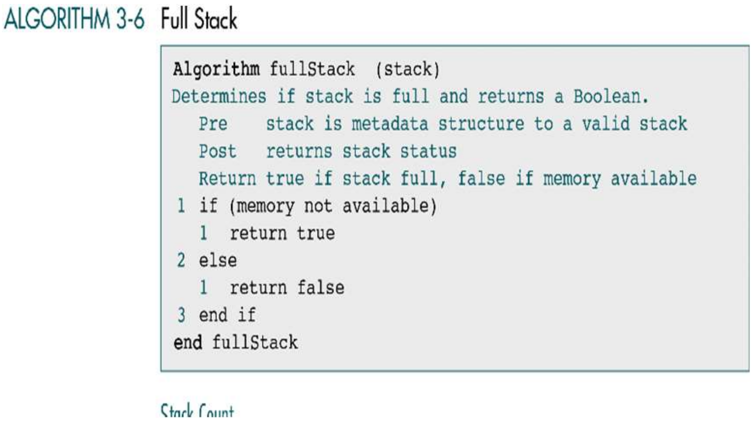


FIGURE 3-10 Pop Stack Example

ALGORITHM 3-4 Stack Top Pseudocode



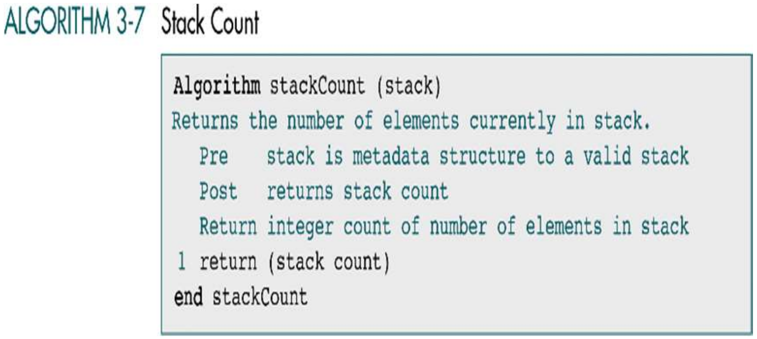
```
Algorithm stackTop  (stack, dataOut)
This algorithm retrieves the data from the top of the stack
without changing the stack.
   Pre    stack is metadata structure to a valid stack
          dataOut is reference variable to receive data
   Post   Data have been returned to calling algorithm
   Return true if data returned, false if underflow
1  if (stack empty)
   1  set success to false
2  else
   1  set dataOut to data in top node
   2  set success to true
3  end if
4  return success
end stackTop
```



ALGORITHM 3-6 Full Stack

```
Algorithm fullStack  (stack)
Determines if stack is full and returns a Boolean.
   Pre    stack is metadata structure to a valid stack
   Post   returns stack status
   Return true if stack full, false if memory available
1 if (memory not available)
   1  return true
2 else
   1  return false
3 end if
end fullStack
```

Stack Count



ALGORITHM 3-7 Stack Count

```
Algorithm stackCount (stack)
Returns the number of elements currently in stack.
   Pre    stack is metadata structure to a valid stack
   Post   returns stack count
   Return integer count of number of elements in stack
1 return (stack count)
end stackCount
```

3-3 C Language Implementations

This section presents a simple non-ADT implementation of a stack. We develop a simple program that inserts random characters into the stack and then prints them.

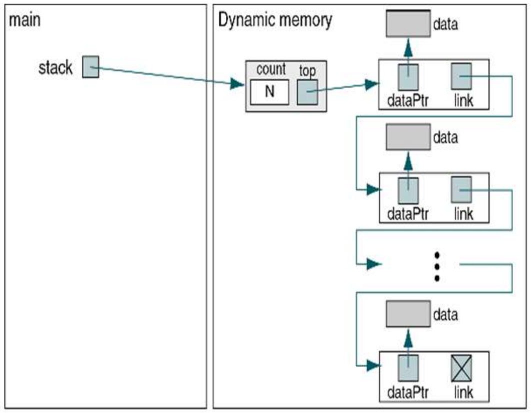
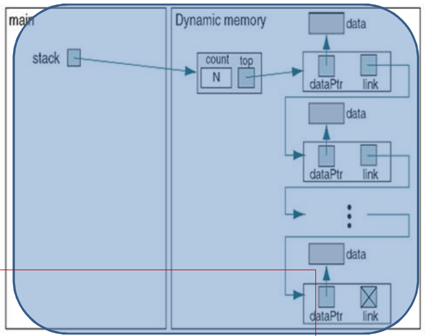


FIGURE 3-12 Stack ADT Structural Concepts

```
#include <stdio.h>
#include <stdlib.h>
#include "stacksADT.h"

int main (void)
{
// Local Definitions
   int* dataPtr ;

   ...

   return 0;
}         // main
```



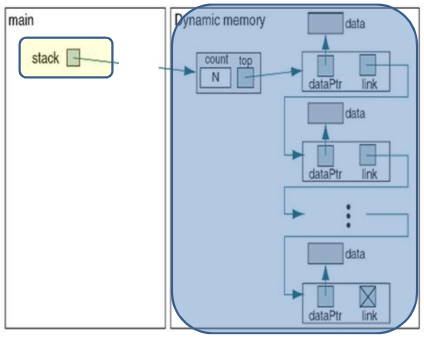
stackADT.h

```
#include <stdlib.h>
#include <stdbool.h>

#include "P3-06.h"  /* Stack ADT Definitions */

//              ADT Prototype Declarations
STACK* createStack  (void);
bool   pushStack    (STACK* stack, void* dataInPtr);
void*  popStack     (STACK* stack);
void*  stackTop     (STACK* stack);
bool   emptyStack   (STACK* stack);
bool   fullStack    (STACK* stack);
int    stackCount   (STACK* stack);
STACK* destroyStack (STACK* stack);

#include "P3-07.h"                           /* Create Stack */
#include "P3-08.h"                           /* Push Stack */
#include "P3-09.h"                           /* Pop Stack */
#include "P3-10.h"                           /* Retrieve Stack Top */
#include "P3-11.h"                           /* Empty Stack */
#include "P3-12.h"                           /* Full Stack */
#include "P3-13.h"                           /* Stack Count */
#include "P3-14.h"                           /* DestroyStack */
```

stackADT.h

```
#include <stdlib.h>
#include <stdbool.h>

#include "P3-06.h"                 /* Stack ADT Definitions */

//       ADT Prototype Declarations
STACK* createStack  (void);
bool   pushStack    (STACK* stack, void* dataInPtr);
void*  popStack     (STACK* stack);
void*  stackTop     (STACK* stack);
bool   emptyStack   (STACK* stack);
bool   fullStack    (STACK* stack);
int    stackCount   (STACK* stack);
STACK* destroyStack (STACK* stack);

#include "P3-07.h"                 /* Create Stack */
#include "P3-08.h"                 /* Push Stack */
#include "P3-09.h"                 /* PopStack */
#include "P3-10.h"                 /* Retrieve Stack Top */
#include "P3-11.h"                 /* EmptyStack */
#include "P3-12.h"                 /* FullStack */
#include "P3-13.h"                 /* Stack Count */
#include "P3-14.h"                 /* DestroyStack */
```

```
#include <stdio.h>
#include <stdlib.h>
#include "stacksADT.h"

int main (void)
{
// Local Definitions
   int* dataPtr ;

   STACK* stack ;

   ...

   return 0;
}         // main
```



P3-06.h

```
typedef struct node
{
   void*          dataPtr;
   struct node*   link;
} STACK_NODE;

typedef struct
{
   int            count;
   STACK_NODE*    top;
} STACK;
```

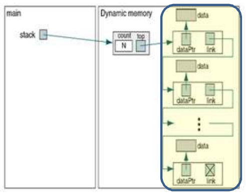
```
#include <stdio.h>
#include <stdlib.h>
#include "stacksADT.h"

int main (void)
{
// Local Definitions
   int* dataPtr ;

   STACK* stack ;

   stack = createStack();
   ...
   return 0;
}         // main
```

Label	Address	Value
dataPtr	326 - 329	
stack	400 - 403	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

P3-07.h

```
STACK* createStack (void)
{
   STACK* stack;
   stack = (STACK*) malloc( sizeof (STACK));
   if (stack)
   {
      stack->count = 0;
      stack->top   = NULL;
   } // if
   return stack;
}        // createStack
```

```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}        // main
```



P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof( STACK_NODE));
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}        // main
```

Label	Address	Value
dataPtr	326 - 329	10210
stack	400 - 403	10100
i	404 - 407	2
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
	…	
	…	
	…	
	…	
	…	
	…	
	…	

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}        // main
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	2
	…	
	…	
	…	
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	
	…	
	…	
	…	
	…	
	…	
	…	

P3-08.h

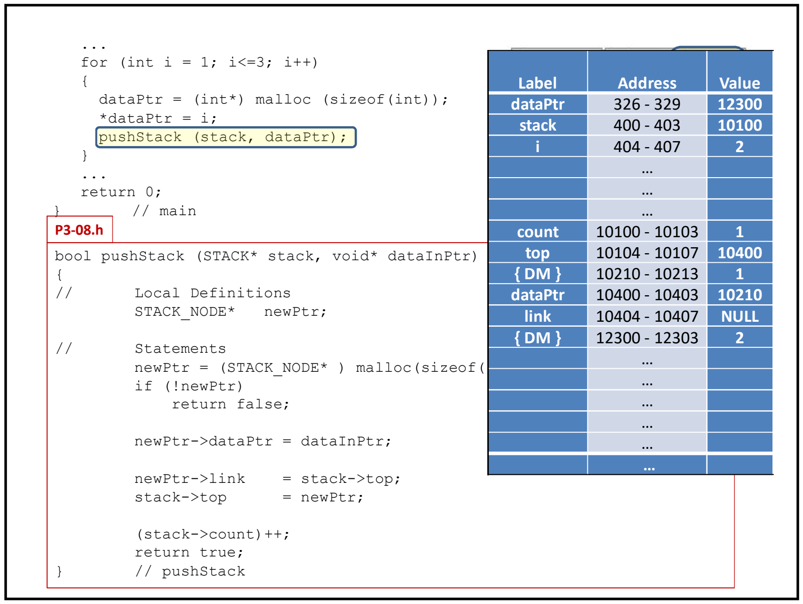
```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}        // main
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	2
	…	
	…	
	…	
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
	…	
	…	
	…	
	…	
	…	
	…	

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

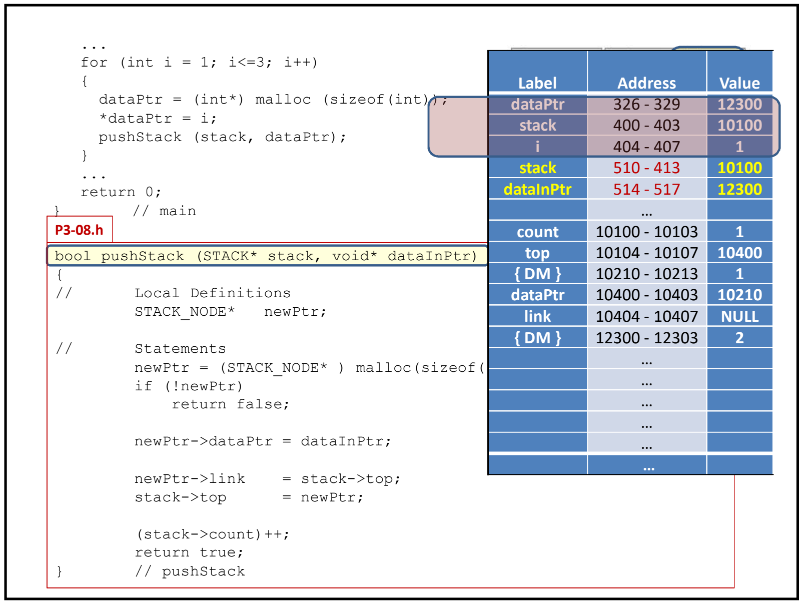


```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link    = stack->top;
         stack->top      = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	2
	…	
	…	
	…	
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
	…	
	…	
	…	
	…	
	…	
	…	

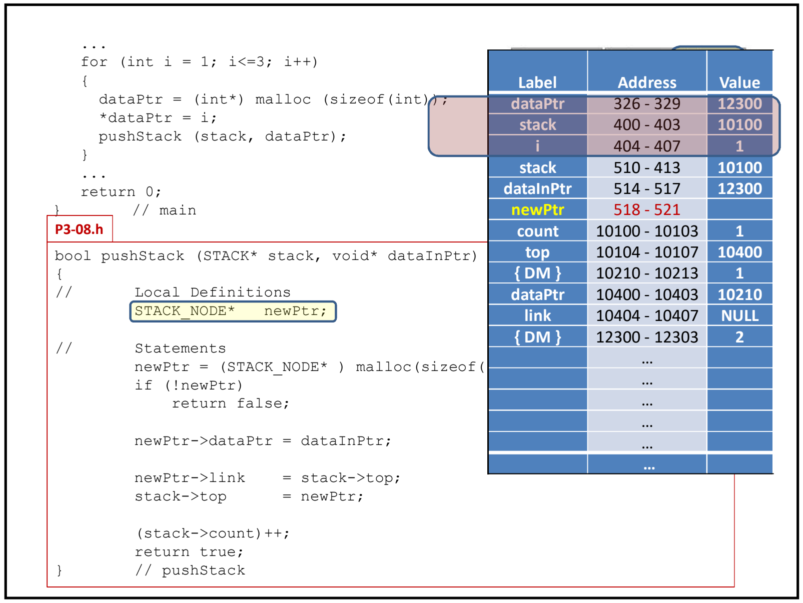


```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link    = stack->top;
         stack->top      = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
	…	
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
	…	
	…	
	…	
	…	
	…	
	…	

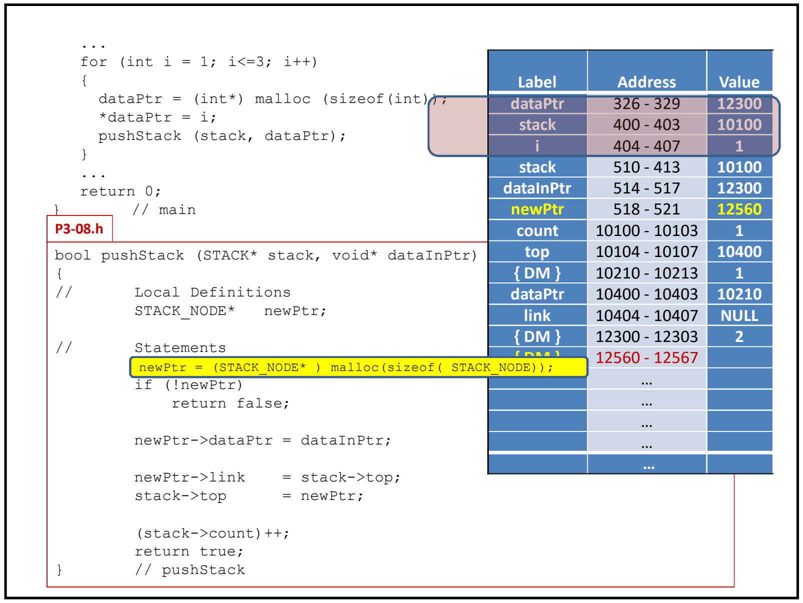


```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link    = stack->top;
         stack->top      = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
	…	
	…	
	…	
	…	
	…	
	…	

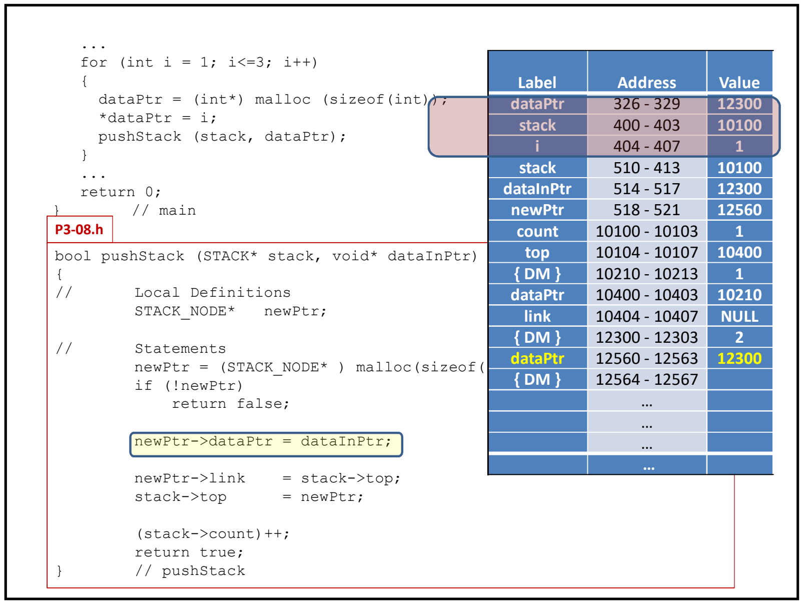


```
    ...
    for (int i = 1; i<=3; i++)
    {
      dataPtr = (int*) malloc (sizeof(int));
      *dataPtr = i;
      pushStack (stack, dataPtr);
    }
    ...
    return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof( STACK_NODE));
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link    = stack->top;
         stack->top      = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	12560
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
{ DM }	12560 - 12567	
	…	
	…	
	…	
	…	
	…	

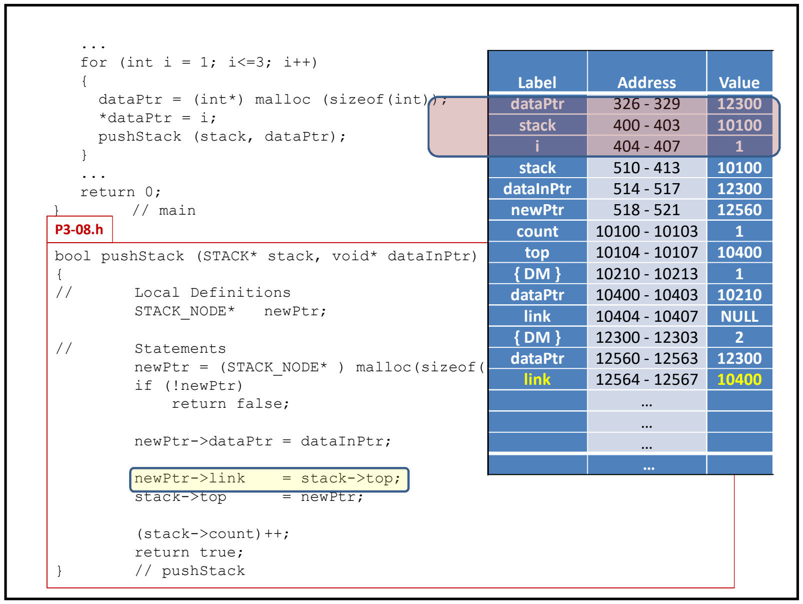


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}        // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	12560
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
{ DM }	12564 - 12567	
	…	
	…	
	…	
	…	

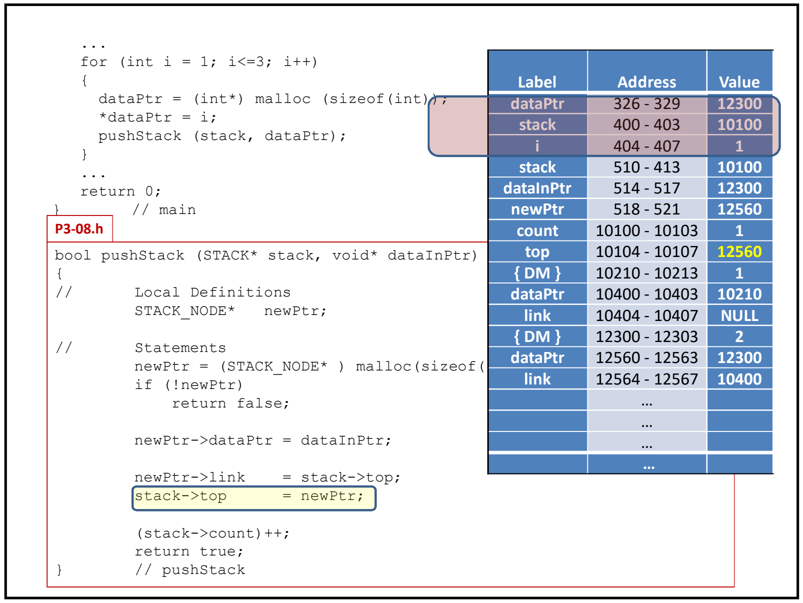


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}        // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	12560
count	10100 - 10103	1
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	…	
	…	
	…	
	…	

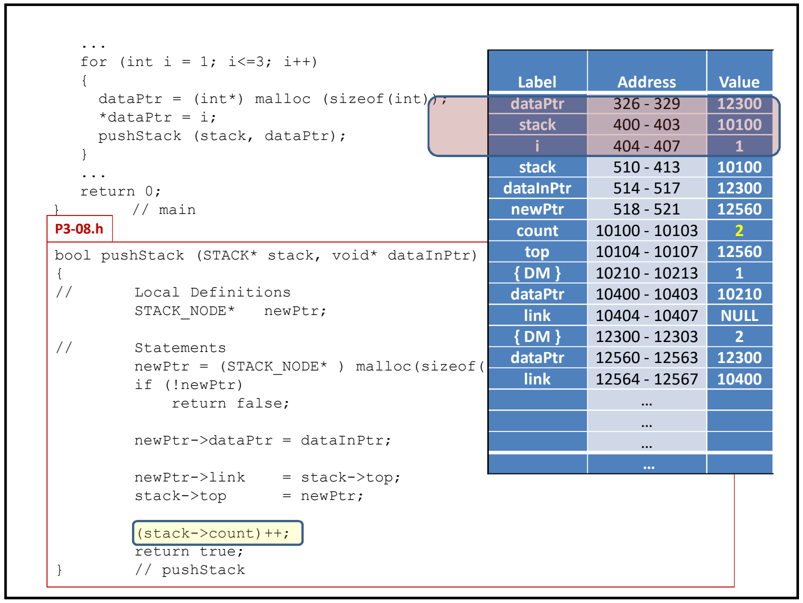


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}        // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	12560
count	10100 - 10103	1
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	…	
	…	
	…	
	…	

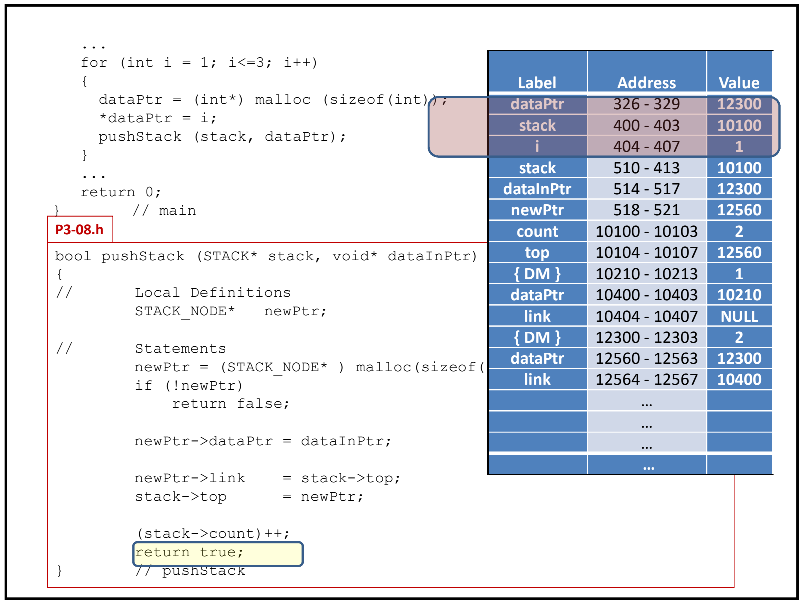


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}        // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//       Local Definitions
         STACK_NODE*    newPtr;

//       Statements
         newPtr = (STACK_NODE* ) malloc(sizeof(
         if (!newPtr)
             return false;

         newPtr->dataPtr = dataInPtr;

         newPtr->link     = stack->top;
         stack->top       = newPtr;

         (stack->count)++;
         return true;
}        // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	12560
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	…	
	…	
	…	
	…	

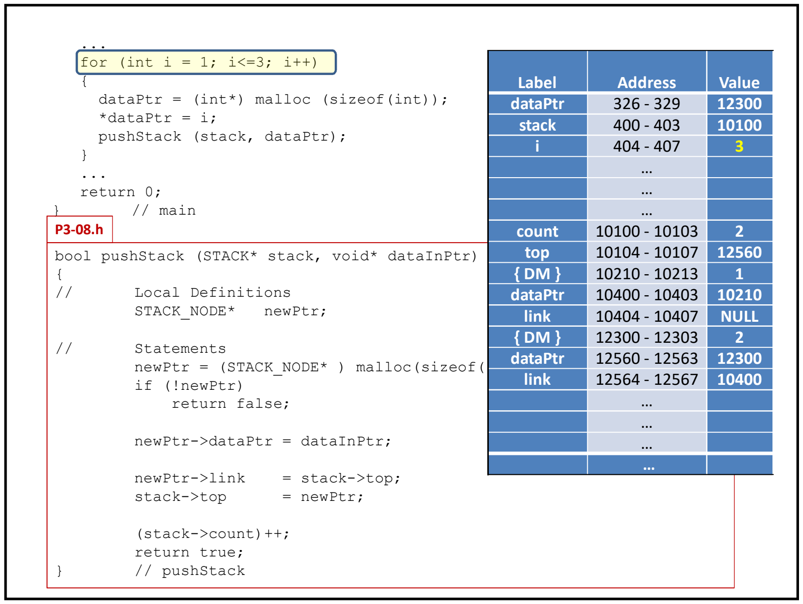


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//        Local Definitions
          STACK_NODE*   newPtr;

//        Statements
          newPtr = (STACK_NODE* ) malloc(sizeof(
          if (!newPtr)
              return false;

          newPtr->dataPtr = dataInPtr;

          newPtr->link    = stack->top;
          stack->top      = newPtr;

          (stack->count)++;
          return true;
}         // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	1
stack	510 - 413	10100
dataInPtr	514 - 517	12300
newPtr	518 - 521	12560
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	…	
	…	
	…	
	…	

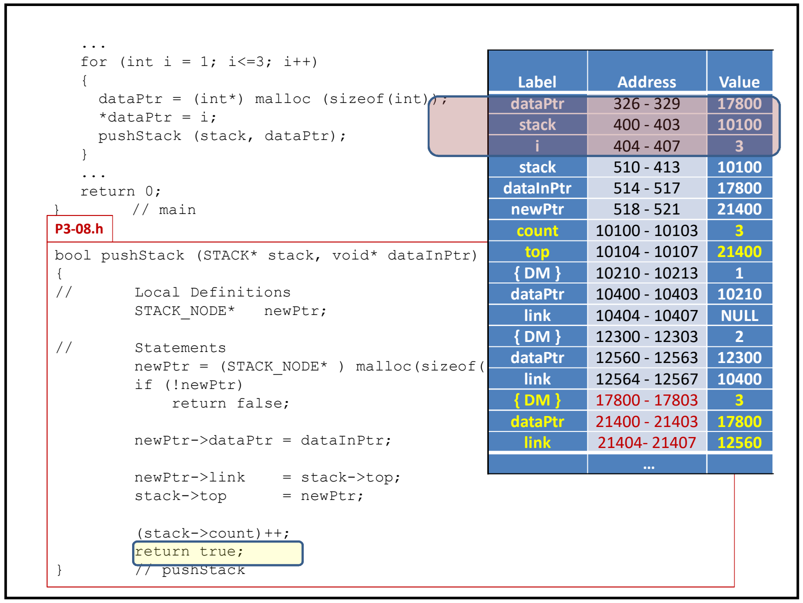


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//        Local Definitions
          STACK_NODE*   newPtr;

//        Statements
          newPtr = (STACK_NODE* ) malloc(sizeof(
          if (!newPtr)
              return false;

          newPtr->dataPtr = dataInPtr;

          newPtr->link    = stack->top;
          stack->top      = newPtr;

          (stack->count)++;
          return true;
}         // pushStack
```

Label	Address	Value
dataPtr	326 - 329	12300
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	…	
	…	
	…	
	…	

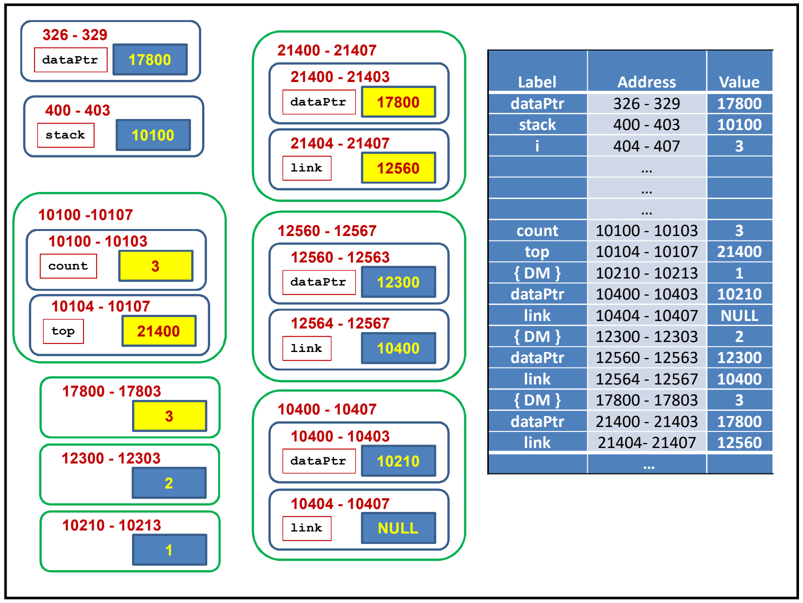


```
   ...
   for (int i = 1; i<=3; i++)
   {
     dataPtr = (int*) malloc (sizeof(int));
     *dataPtr = i;
     pushStack (stack, dataPtr);
   }
   ...
   return 0;
}          // main
```

P3-08.h

```
bool pushStack (STACK* stack, void* dataInPtr)
{
//        Local Definitions
          STACK_NODE*   newPtr;

//        Statements
          newPtr = (STACK_NODE* ) malloc(sizeof(
          if (!newPtr)
              return false;

          newPtr->dataPtr = dataInPtr;

          newPtr->link    = stack->top;
          stack->top      = newPtr;

          (stack->count)++;
          return true;
}         // pushStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	510 - 413	10100
dataInPtr	514 - 517	17800
newPtr	518 - 521	21400
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	



326 - 329
dataPtr 17800

400 - 403
stack 10100

10100 -10107
10100 - 10103
count 3
10104 - 10107
top 21400

17800 - 17803
3

12300 - 12303
2

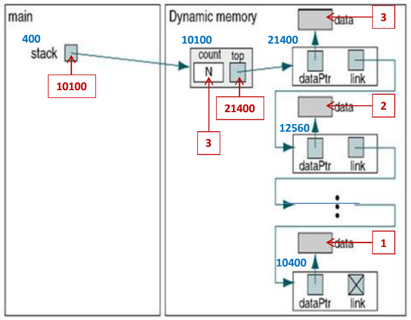
10210 - 10213
1

21400 - 21407
21400 - 21403
dataPtr 17800
21404 - 21407
link 12560

12560 - 12567
12560 - 12563
dataPtr 12300
12564 - 12567
link 10400

10400 - 10407
10400 - 10403
dataPtr 10210
10404 - 10407
link NULL

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	



Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

ALGORITHM 3-3 Pop Stack

```
Algorithm popStack (stack, dataOut)
This algorithm pops the item on the top of the stack and
returns it to the user.
   Pre    stack passed by reference
          dataOut is reference variable to receive data
   Post   Data have been returned to calling algorithm
   Return true if successful; false if underflow
1  if (stack empty)
   1  set success to false
2  else
   1  set dataOut to data in top node
   2  make second node the top node
   3  decrement stack count
   4  set success to true
3  end if
4  return success
end popStack
```

```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}         // main
```

P3-11.h

```
bool emptyStack (STACK* stack)
{
// Statements
   return (stack->count == 0);
}  // emptyStack
```

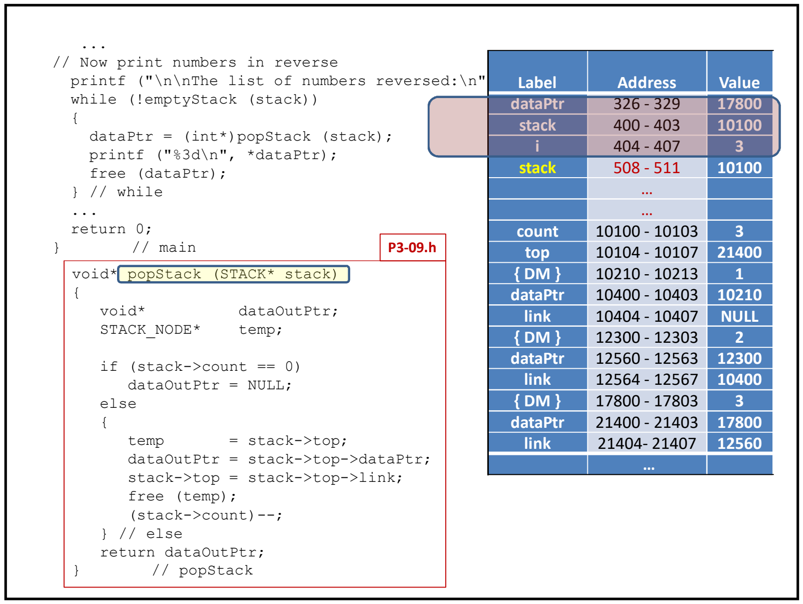
Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}         // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}         // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

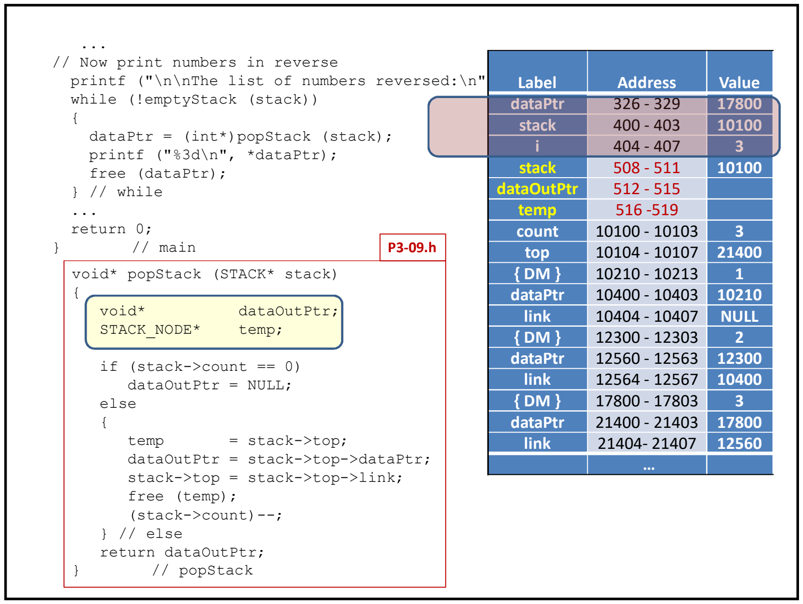


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
	...	
	...	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

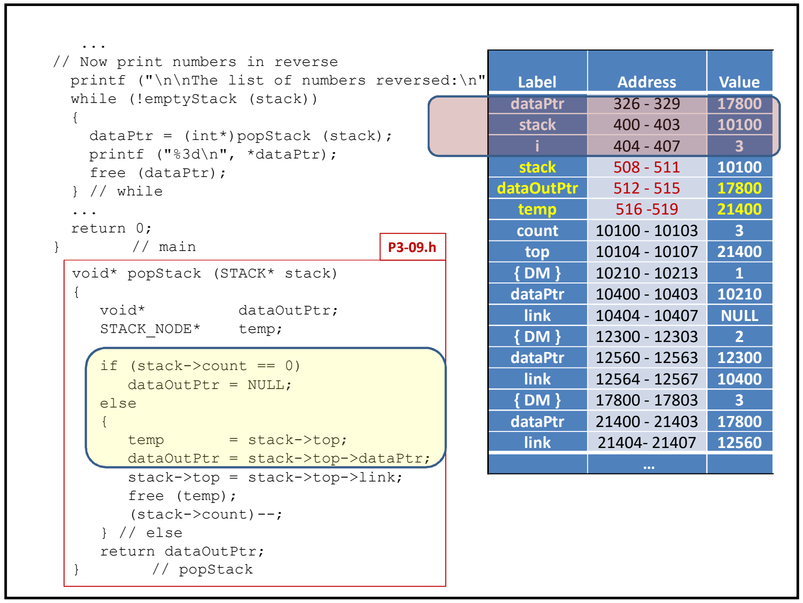


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
dataOutPtr	512 - 515	
temp	516 -519	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

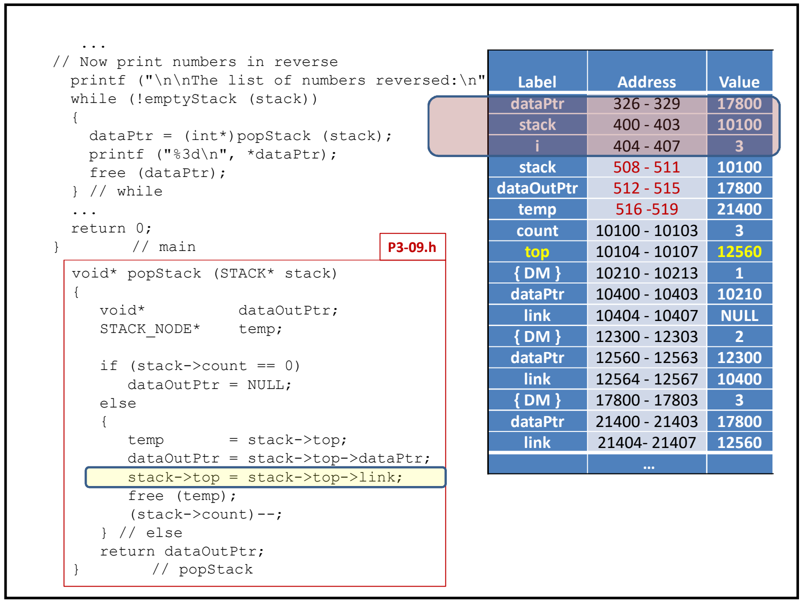


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
dataOutPtr	512 - 515	17800
temp	516 -519	21400
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

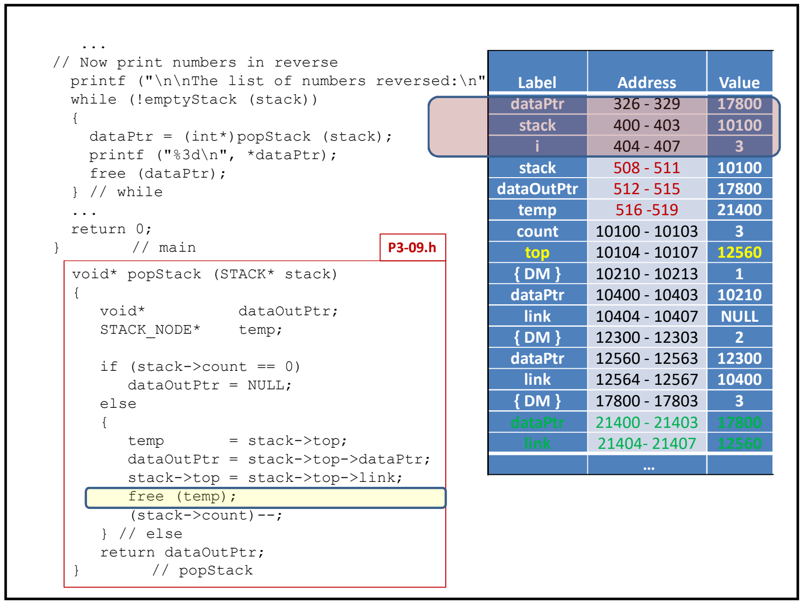


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
dataOutPtr	512 - 515	17800
temp	516 -519	21400
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

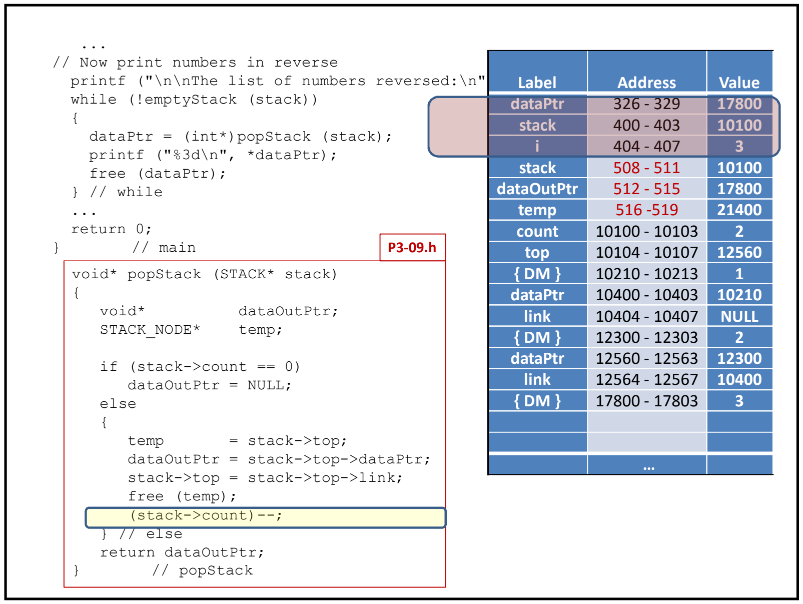


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
dataOutPtr	512 - 515	17800
temp	516 -519	21400
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

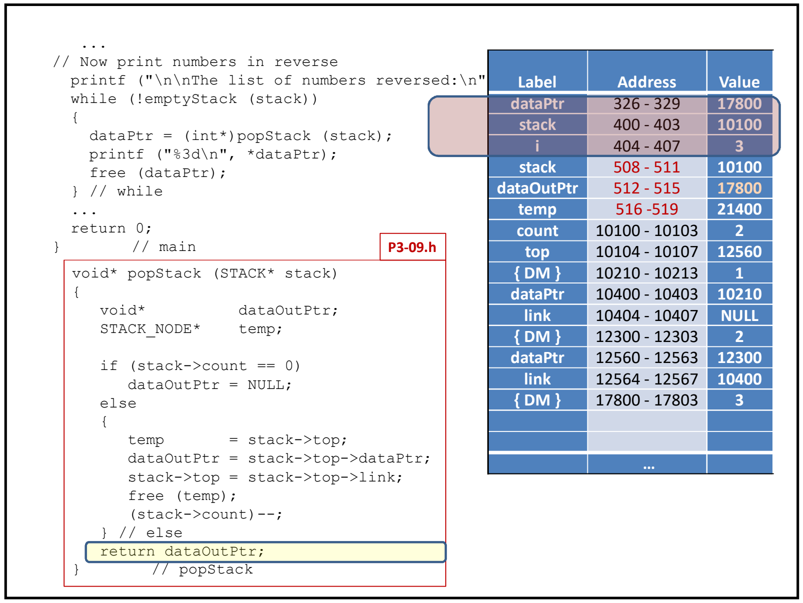


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
dataOutPtr	512 - 515	17800
temp	516 -519	21400
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
	...	

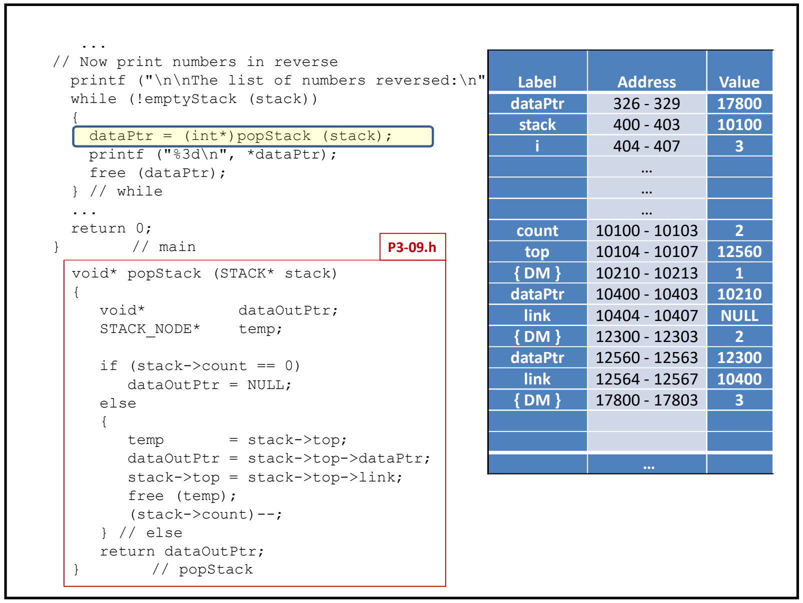


```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
dataOutPtr	512 - 515	17800
temp	516 -519	21400
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
	...	



```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}        // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp       = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}        // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	...	
	...	
	...	
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
	...	

```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}         // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp        = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}         // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
	…	

```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}         // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp        = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}         // popStack
```

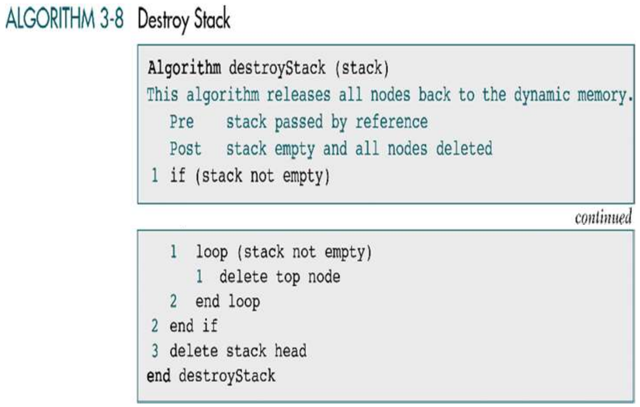
Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
count	10100 - 10103	2
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	…	

```
   ...
// Now print numbers in reverse
  printf ("\n\nThe list of numbers reversed:\n"
  while (!emptyStack (stack))
  {
    dataPtr = (int*)popStack (stack);
    printf ("%3d\n", *dataPtr);
    free (dataPtr);
  } // while
  ...
  return 0;
}         // main
```

P3-09.h

```
void* popStack (STACK* stack)
{
   void*          dataOutPtr;
   STACK_NODE*    temp;

   if (stack->count == 0)
      dataOutPtr = NULL;
   else
   {
      temp        = stack->top;
      dataOutPtr = stack->top->dataPtr;
      stack->top = stack->top->link;
      free (temp);
      (stack->count)--;
   } // else
   return dataOutPtr;
}         // popStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	

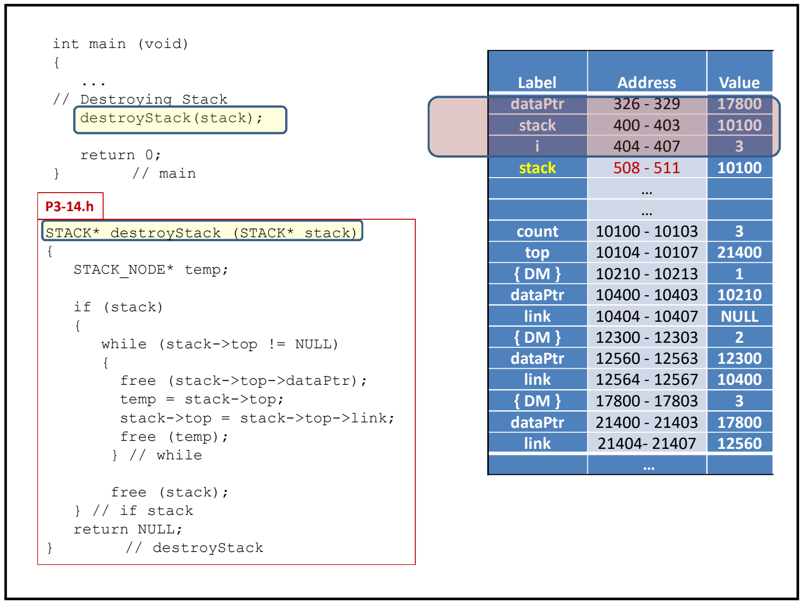


ALGORITHM 3-8 Destroy Stack

```
Algorithm destroyStack (stack)
This algorithm releases all nodes back to the dynamic memory.
  Pre    stack passed by reference
  Post   stack empty and all nodes deleted
1 if (stack not empty)
```

continued

```
   1  loop (stack not empty)
      1  delete top node
   2  end loop
2 end if
3 delete stack head
end destroyStack
```

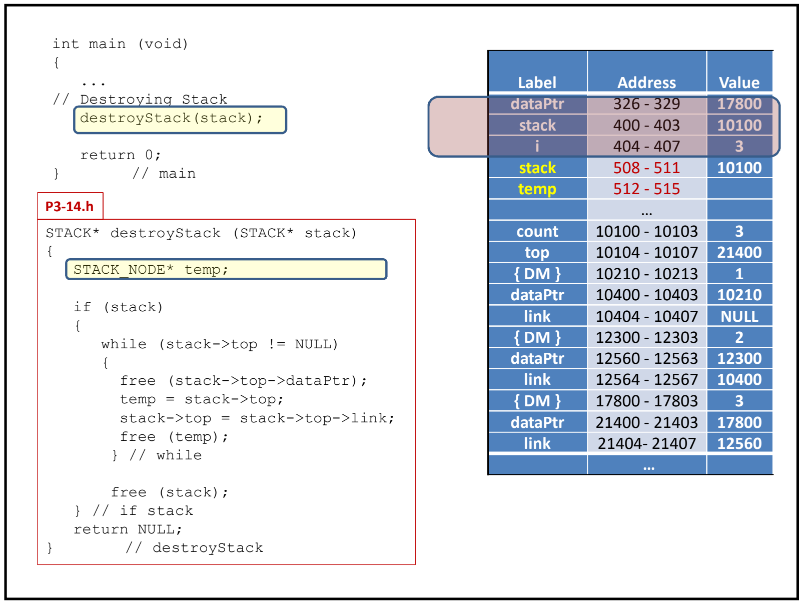


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
	...	
	...	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

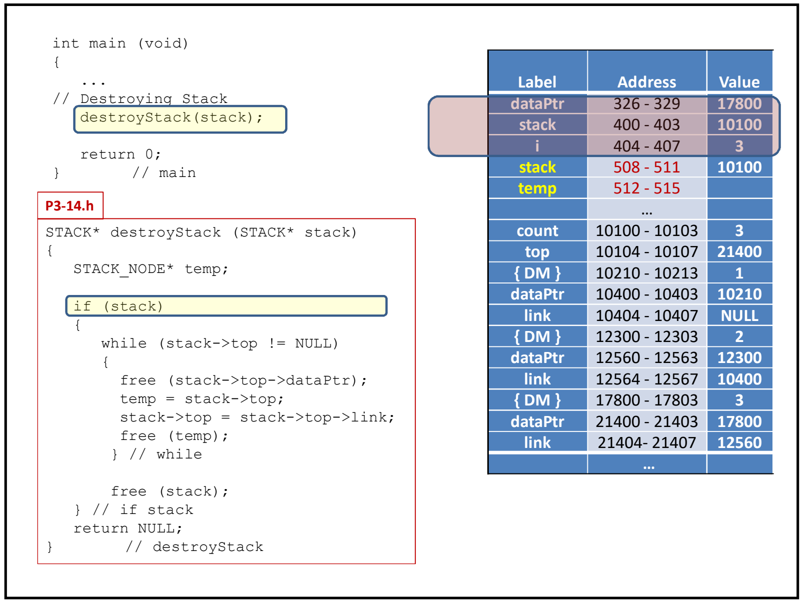


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	
	...	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

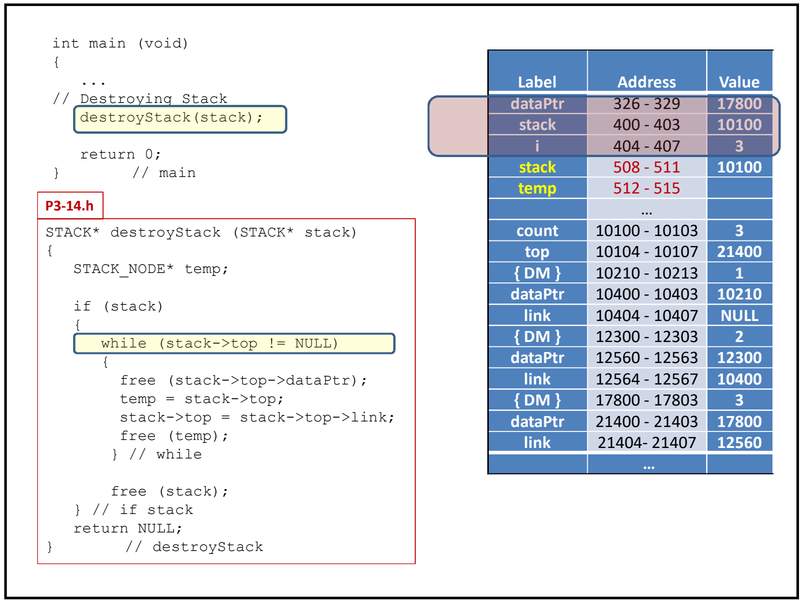


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	
	...	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

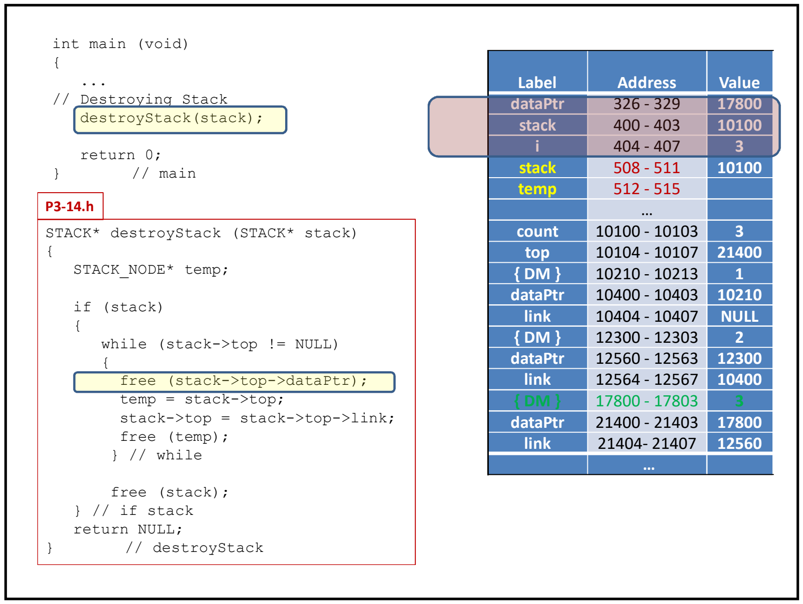


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	
	...	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	...	

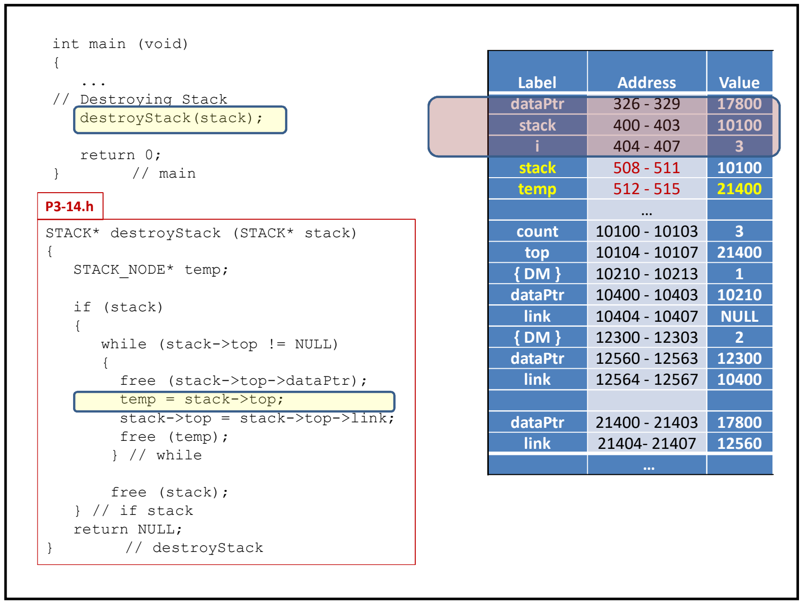


```
int main (void)
{
    ...
// Destroying Stack
   destroyStack(stack);

    return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
    STACK_NODE* temp;

    if (stack)
    {
       while (stack->top != NULL)
       {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
    } // if stack
    return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	
	…	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
{ DM }	17800 - 17803	3
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

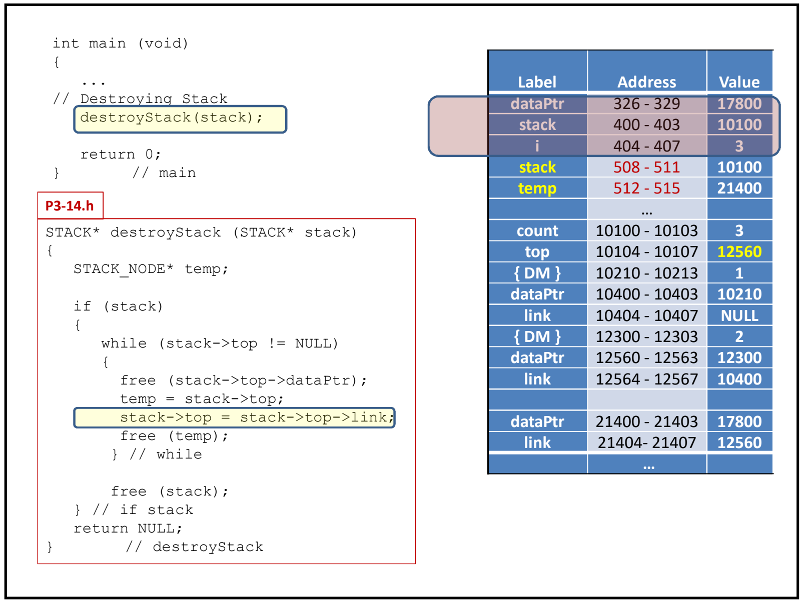


```
int main (void)
{
    ...
// Destroying Stack
   destroyStack(stack);

    return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
    STACK_NODE* temp;

    if (stack)
    {
       while (stack->top != NULL)
       {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
    } // if stack
    return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	21400
	…	
count	10100 - 10103	3
top	10104 - 10107	21400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

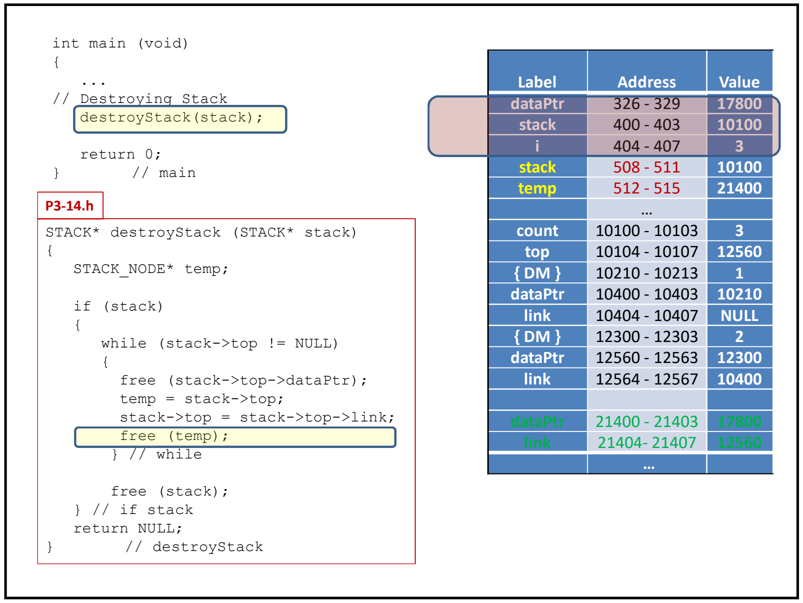


```
int main (void)
{
    ...
// Destroying Stack
   destroyStack(stack);

    return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
    STACK_NODE* temp;

    if (stack)
    {
       while (stack->top != NULL)
       {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
    } // if stack
    return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	21400
	…	
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

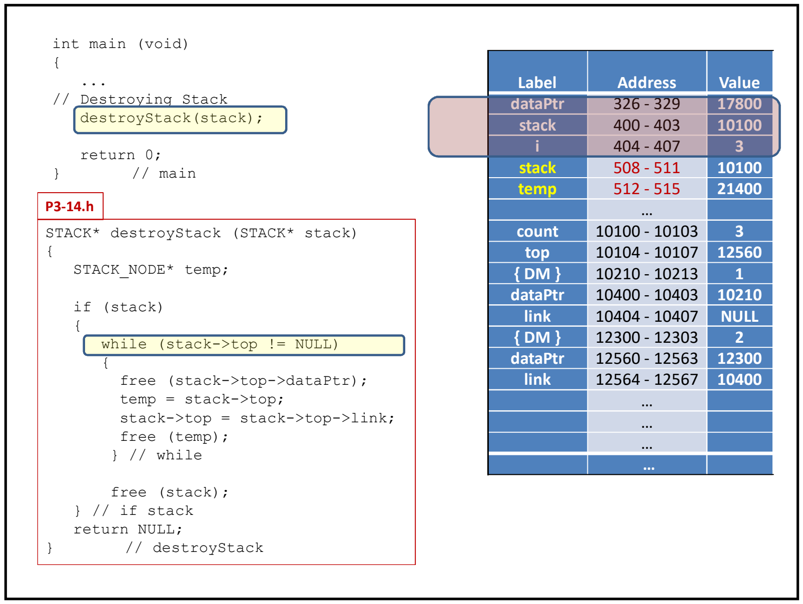


```
int main (void)
{
    ...
// Destroying Stack
   destroyStack(stack);

    return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
    STACK_NODE* temp;

    if (stack)
    {
       while (stack->top != NULL)
       {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
    } // if stack
    return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	21400
	…	
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
dataPtr	21400 - 21403	17800
link	21404- 21407	12560
	…	

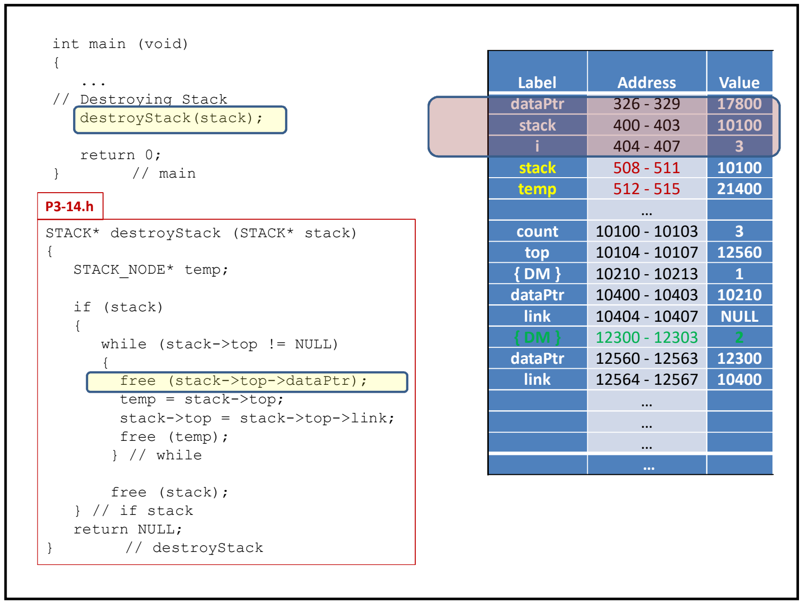


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
        {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	21400
	...	
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	...	
	...	
	...	
	...	

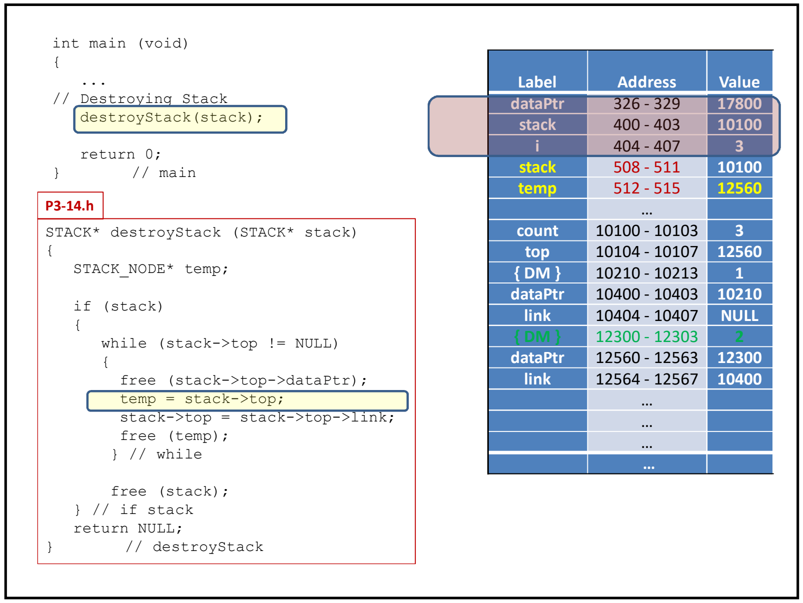


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
        {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	21400
	...	
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	...	
	...	
	...	
	...	

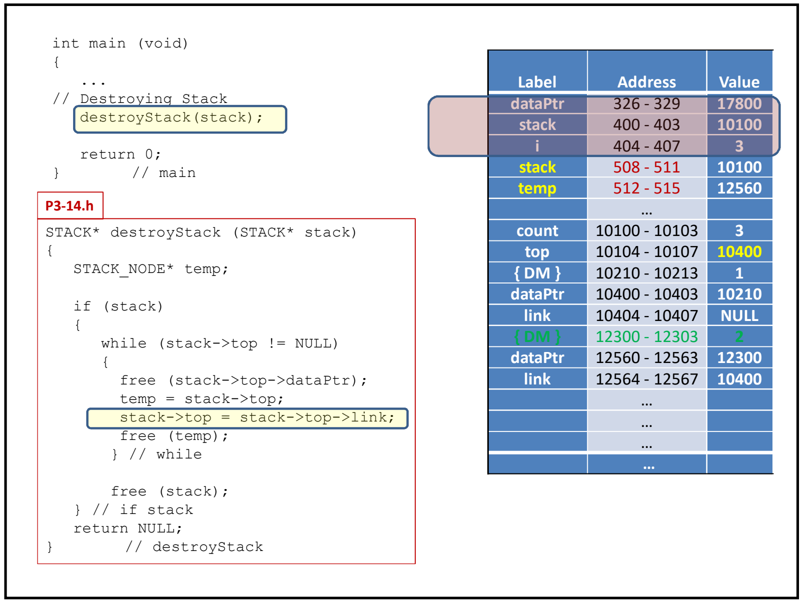


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
        {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	12560
	...	
count	10100 - 10103	3
top	10104 - 10107	12560
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	...	
	...	
	...	
	...	

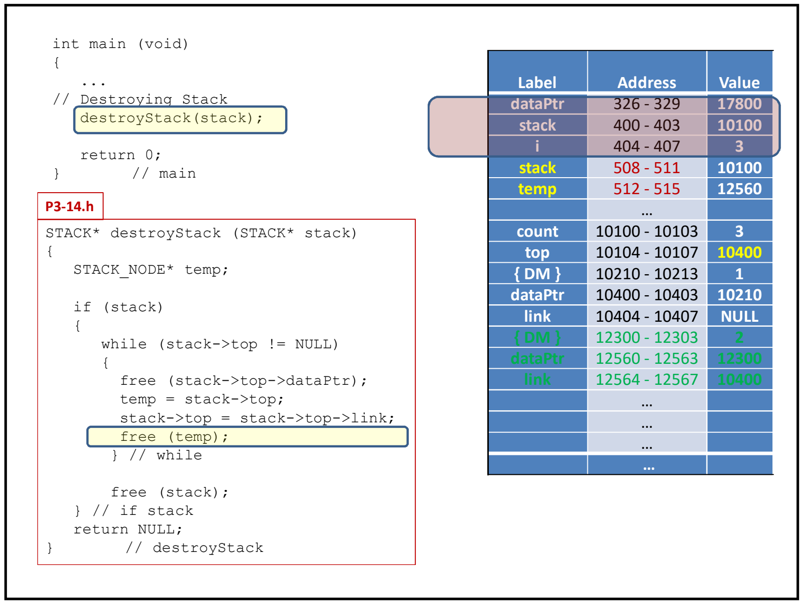


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
        {
         free (stack->top->dataPtr);
         temp = stack->top;
         stack->top = stack->top->link;
         free (temp);
        } // while

        free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	12560
	...	
count	10100 - 10103	3
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	...	
	...	
	...	
	...	

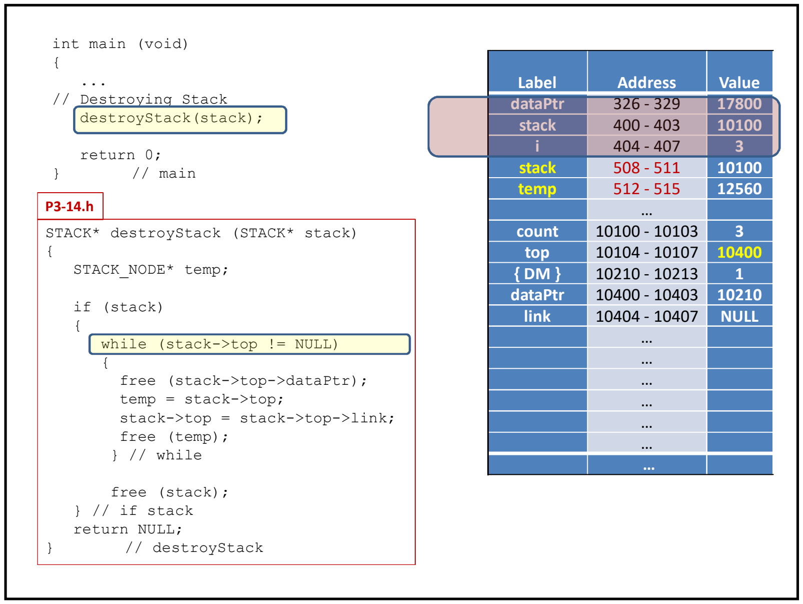


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
      } // while

      free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	12560
	...	
count	10100 - 10103	3
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
{ DM }	12300 - 12303	2
dataPtr	12560 - 12563	12300
link	12564 - 12567	10400
	...	
	...	
	...	
	...	

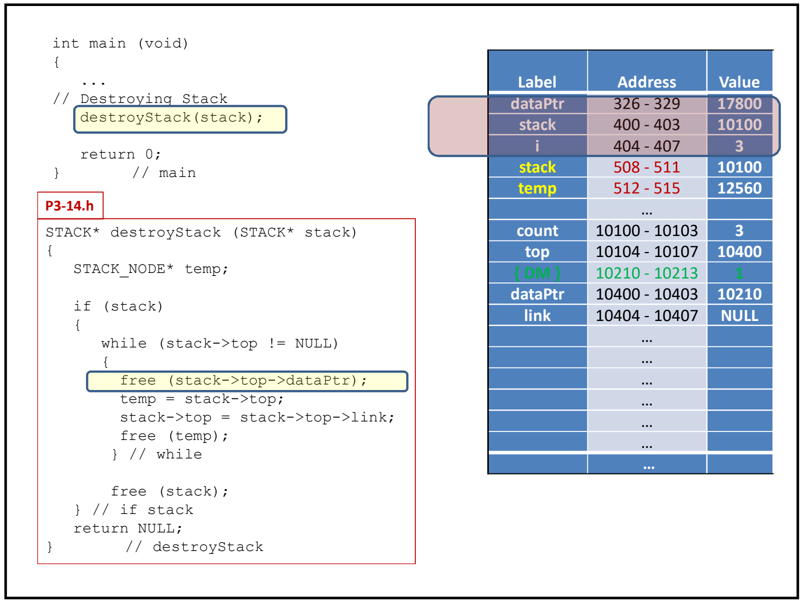


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
      } // while

      free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	12560
	...	
count	10100 - 10103	3
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	

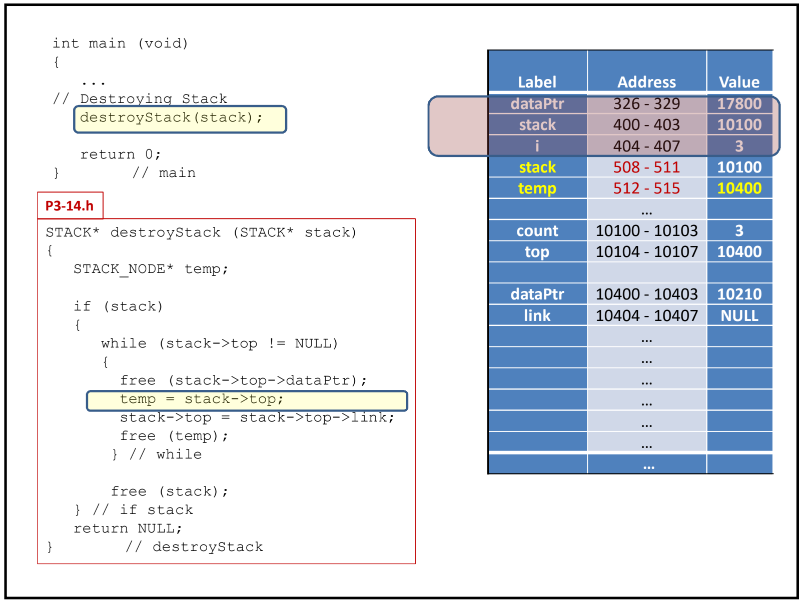


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
      } // while

      free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	12560
	...	
count	10100 - 10103	3
top	10104 - 10107	10400
{ DM }	10210 - 10213	1
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	

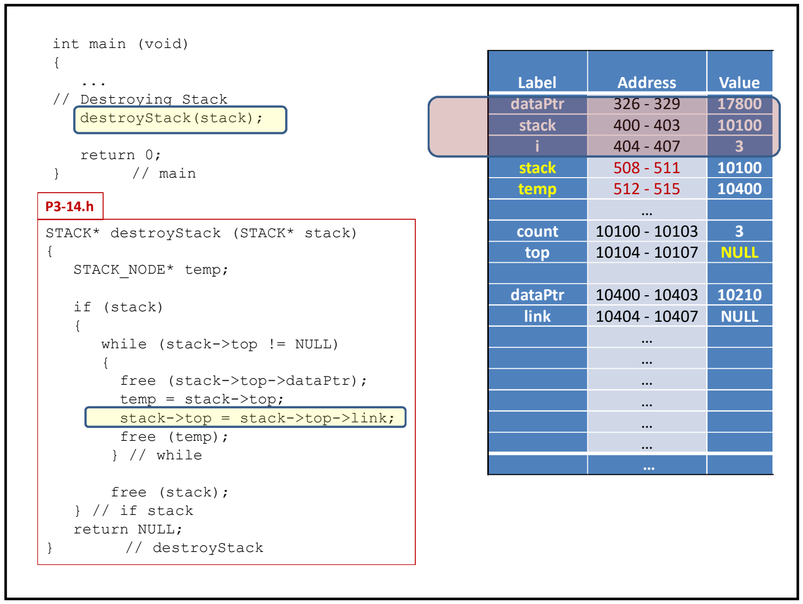


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
      } // while

      free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	10400
	...	
count	10100 - 10103	3
top	10104 - 10107	10400
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	

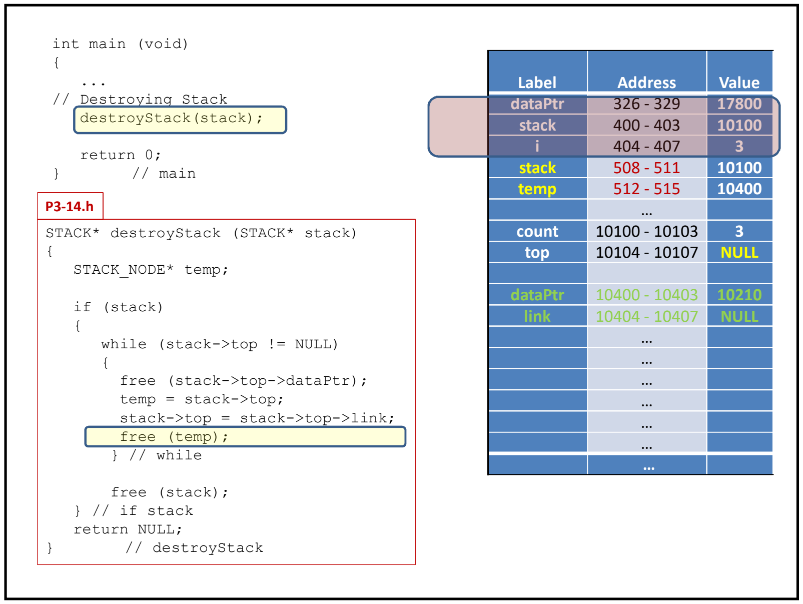


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	10400
	...	
count	10100 - 10103	3
top	10104 - 10107	NULL
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	

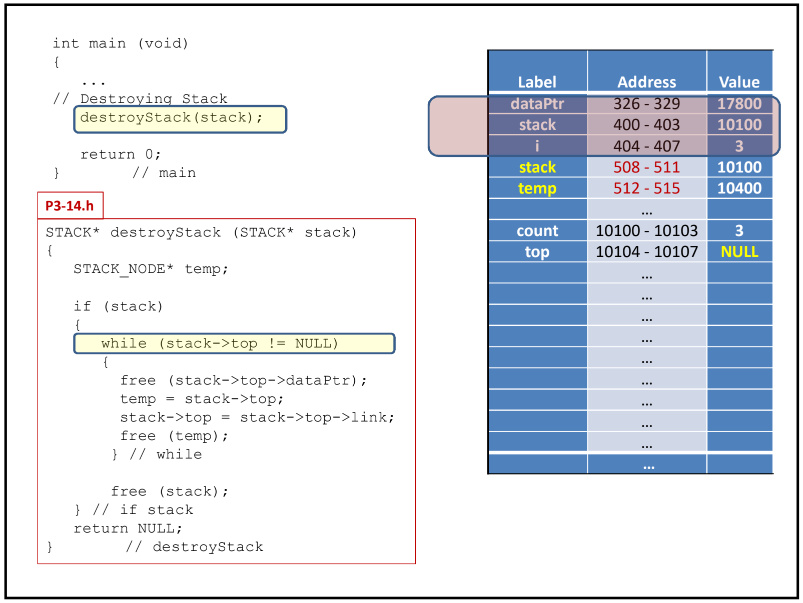


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	10400
	...	
count	10100 - 10103	3
top	10104 - 10107	NULL
dataPtr	10400 - 10403	10210
link	10404 - 10407	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	

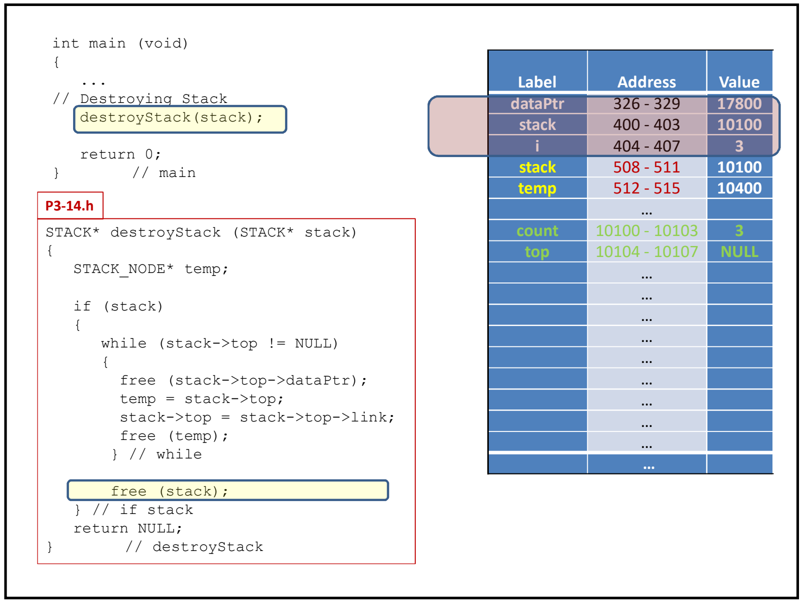


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	10400
	...	
count	10100 - 10103	3
top	10104 - 10107	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

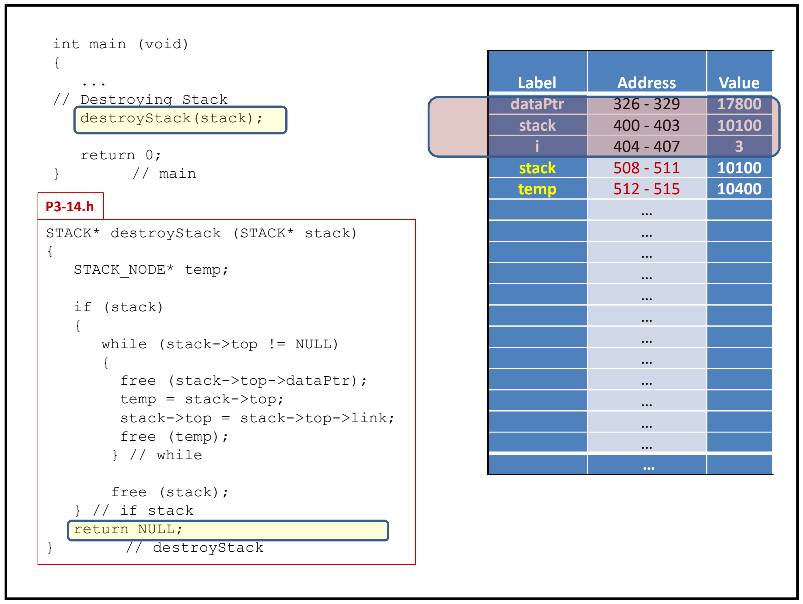


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}         // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}         // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	10400
	...	
count	10100 - 10103	3
top	10104 - 10107	NULL
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	
	...	

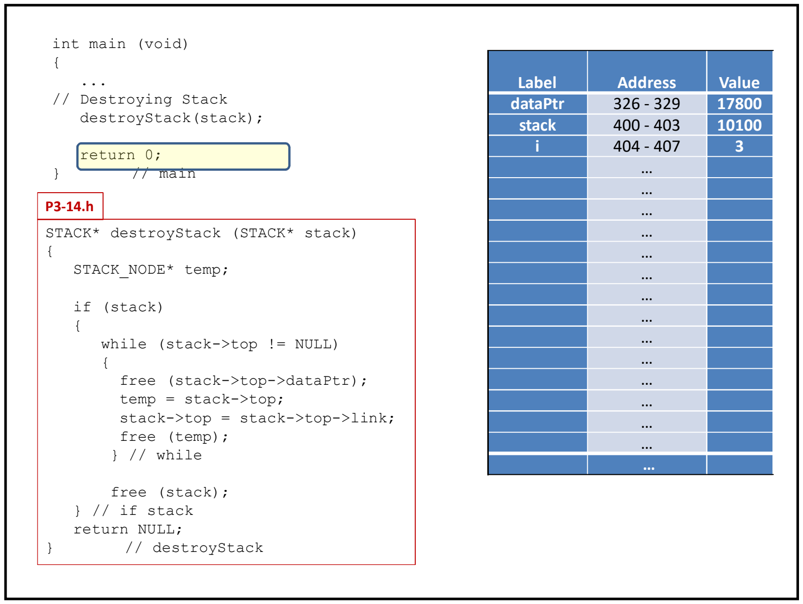


```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
  return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
stack	508 - 511	10100
temp	512 - 515	10400
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	



```
int main (void)
{
   ...
// Destroying Stack
   destroyStack(stack);

   return 0;
}        // main
```

P3-14.h

```
STACK* destroyStack (STACK* stack)
{
   STACK_NODE* temp;

   if (stack)
   {
      while (stack->top != NULL)
      {
        free (stack->top->dataPtr);
        temp = stack->top;
        stack->top = stack->top->link;
        free (temp);
       } // while

       free (stack);
   } // if stack
   return NULL;
}        // destroyStack
```

Label	Address	Value
dataPtr	326 - 329	17800
stack	400 - 403	10100
i	404 - 407	3
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	
	…	