進階C語言實務_期末專案

A customized command line and file system

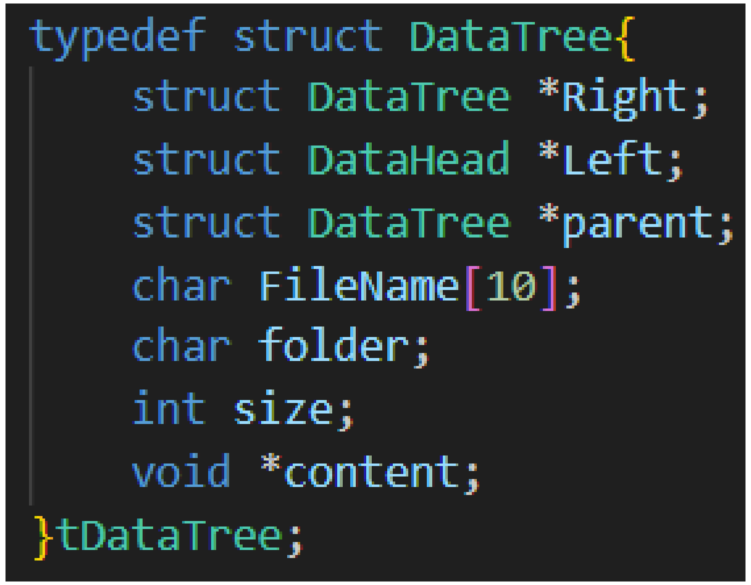
-112368001_電子碩一_廖皓呈

-112368003_電子碩一_高敬偉

-112368017_電子碩一_楊皓麟

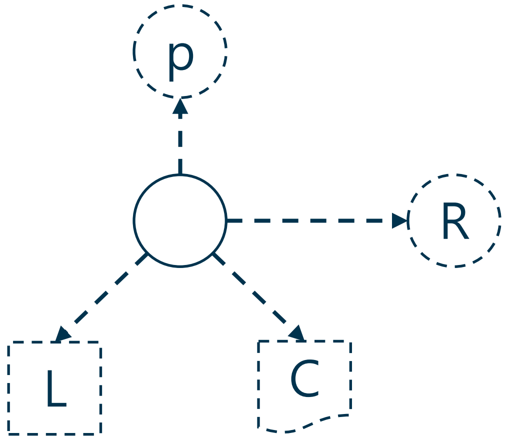
Struct_DataTree

DataTree：資料節點



```
typedef struct DataTree{
    struct DataTree *Right;
    struct DataHead *Left;
    struct DataTree *parent;
    char FileName[10];
    char folder;
    int size;
    void *content;
}tDataTree;
```

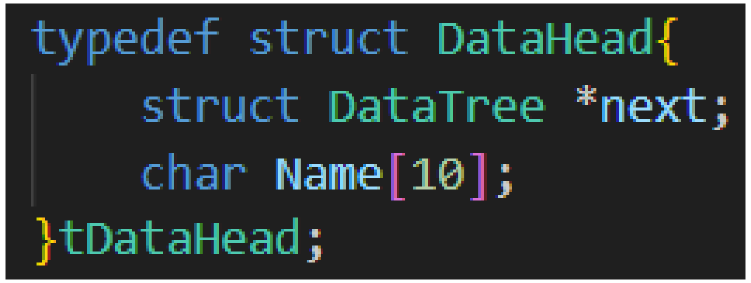
-	*Right	右子樹，定義為同階層之資料節點
-	*parent	父節點，於刪除資料節點時重新鏈結需要
-	*Left	左子樹，只有當該節點為資料夾類型時需要，且指向類型為DataHead
-	FileName	資料節點名稱（資料夾或檔案名稱）
-	folder	旗標，是否為資料夾
-	size	資料檔案大小
-	*content	資料檔案存放位置指標



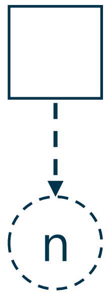
Struct_DataHead

DataHead：各目錄之Head

-	*next	指向該資料夾之第一個資料節點，且指向類型為DataTree
-	Name	根目錄名稱

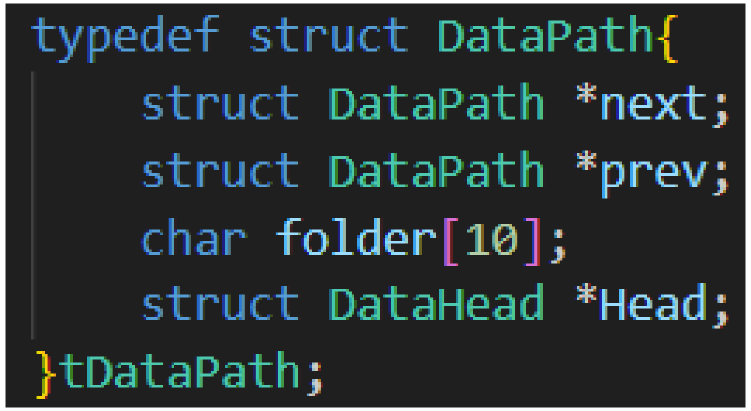


```
typedef struct DataHead{
    struct DataTree *next;
    char Name[10];
}tDataHead;
```



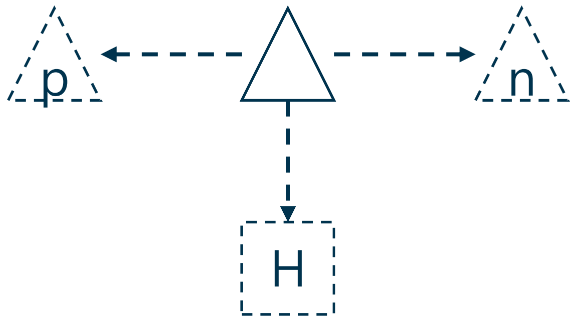
Struct_DataPath

DataPath：存放路徑資訊，各操作需透過此結構進行索引資料，並於cd中維持此結構



```
typedef struct DataPath{
    struct DataPath *next;
    struct DataPath *prev;
    char folder[10];
    struct DataHead *Head;
}tDataPath;
```

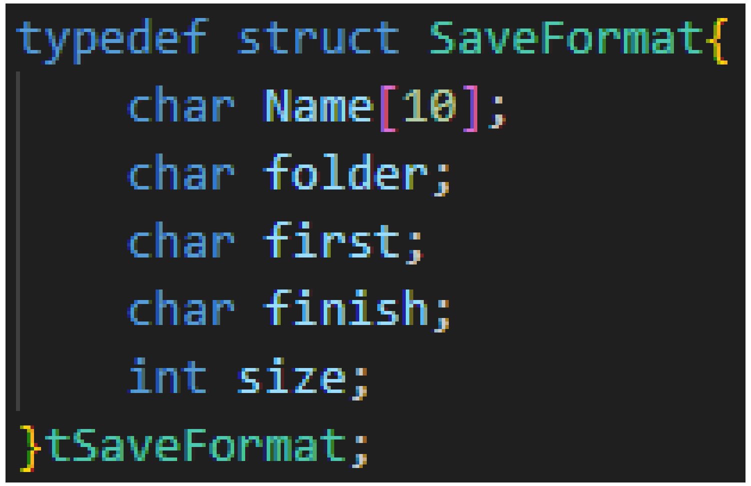
- *next	下一路徑，若cd至子目錄時會增加此路徑；反之減少
- *prev	前一路徑
- *Head	指向個別路徑之Haed，進而由Head索引存放之資料節點，且指向類型為DataHead
- folder	路徑目錄名稱



Struct_SaveFormat

T H P

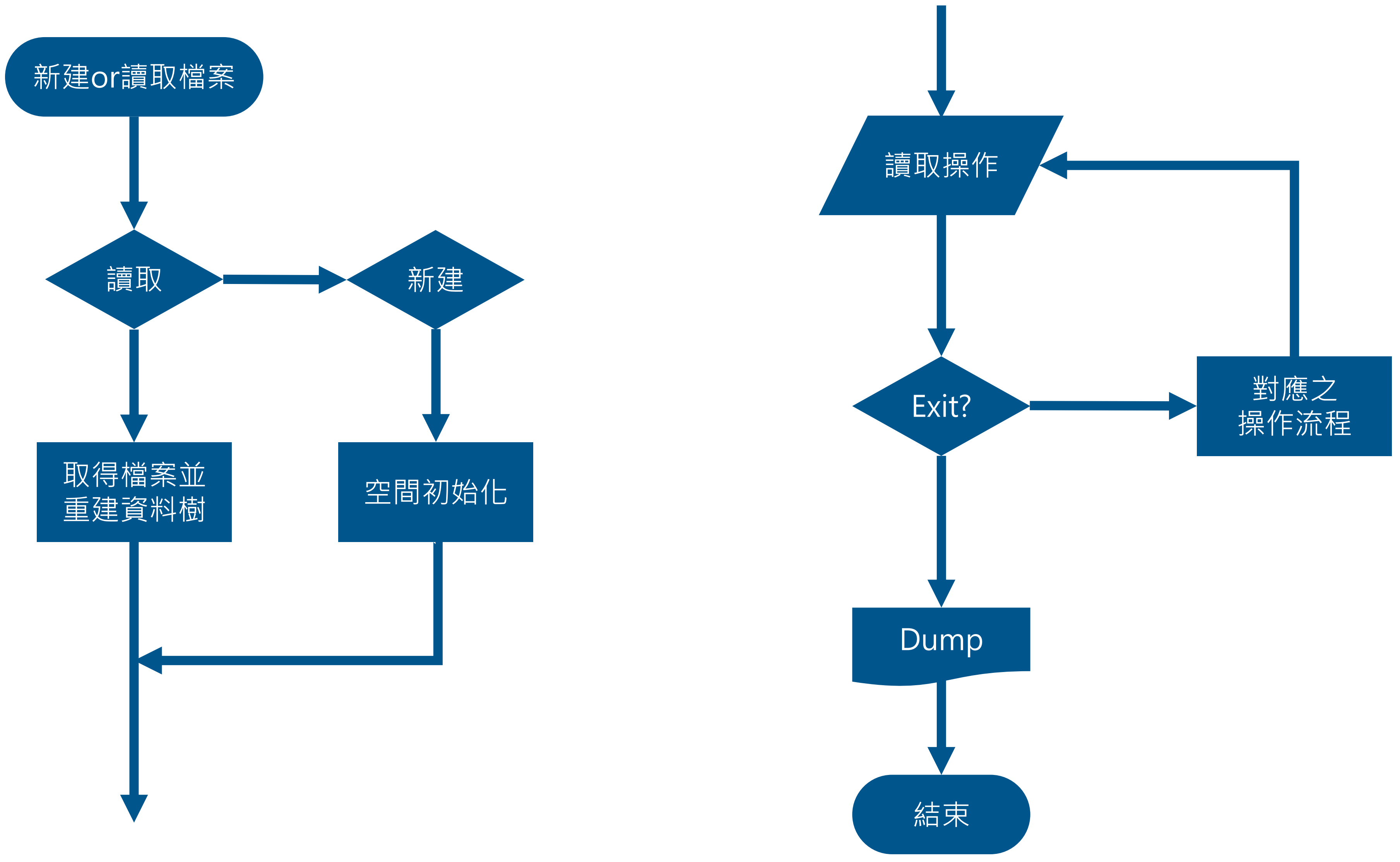
SaveFormat：用於輸出dump的結構



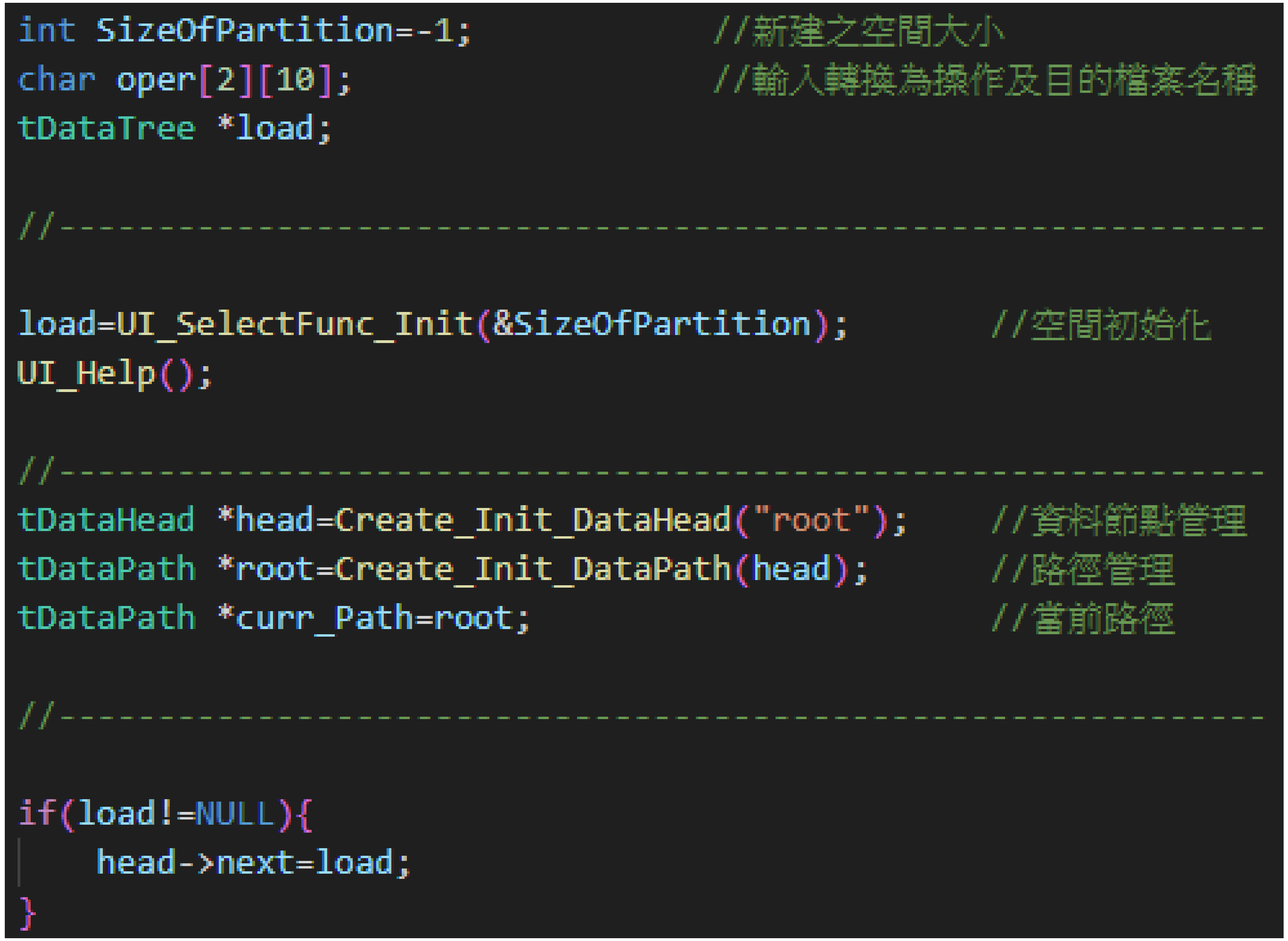
```c
typedef struct SaveFormat{
    char Name[10];
    char folder;
    char first;
    char finish;
    int size;
}tSaveFormat;
```

-	Name	資料節點名稱（資料夾或檔案名稱）
-	folder	是否為資料夾
-	first	是否為該目錄內第一個資料節點
-	finish	是否為該目錄內最後一個資料節點
-	size	檔案=>大小；資料夾=>0表示該目錄內有檔案 1表示該目錄內無檔案

主程式流程

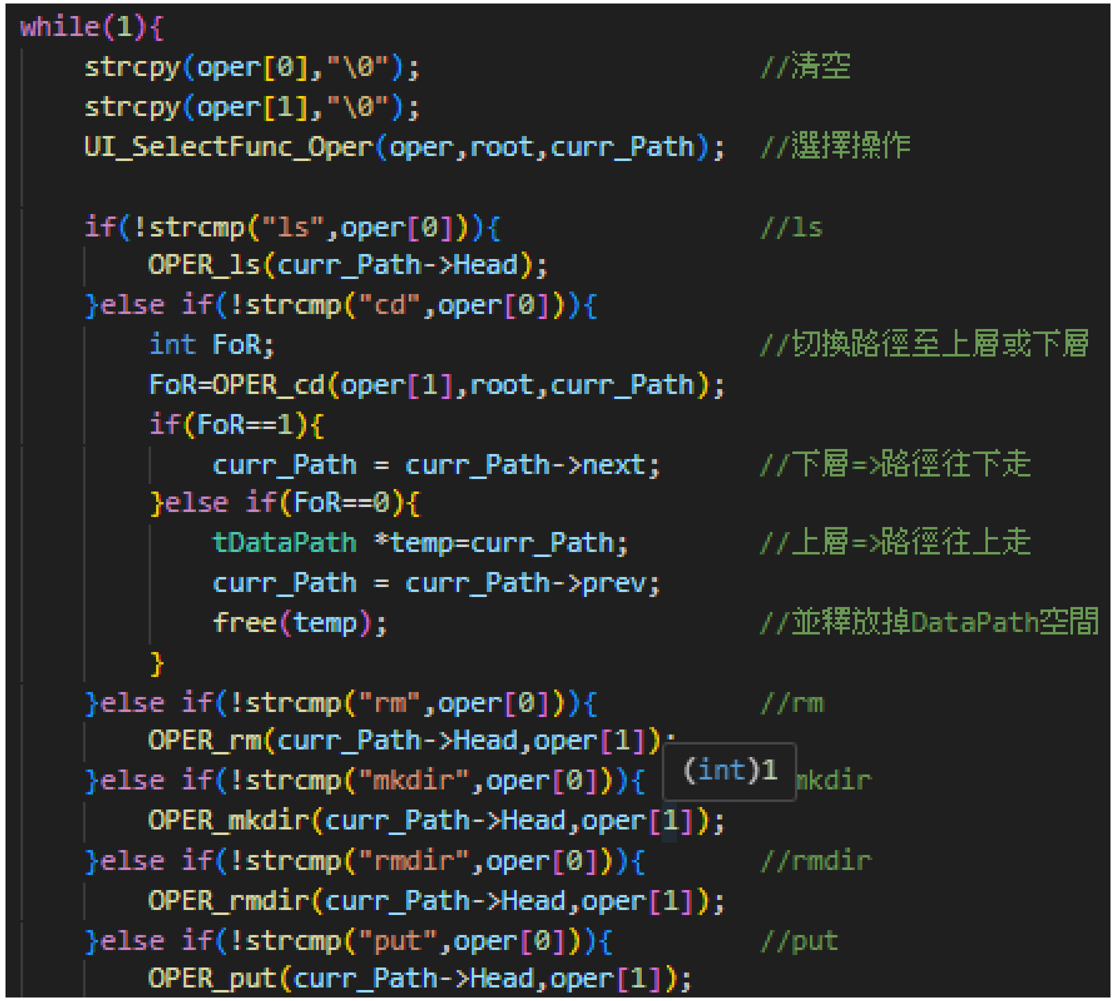


main

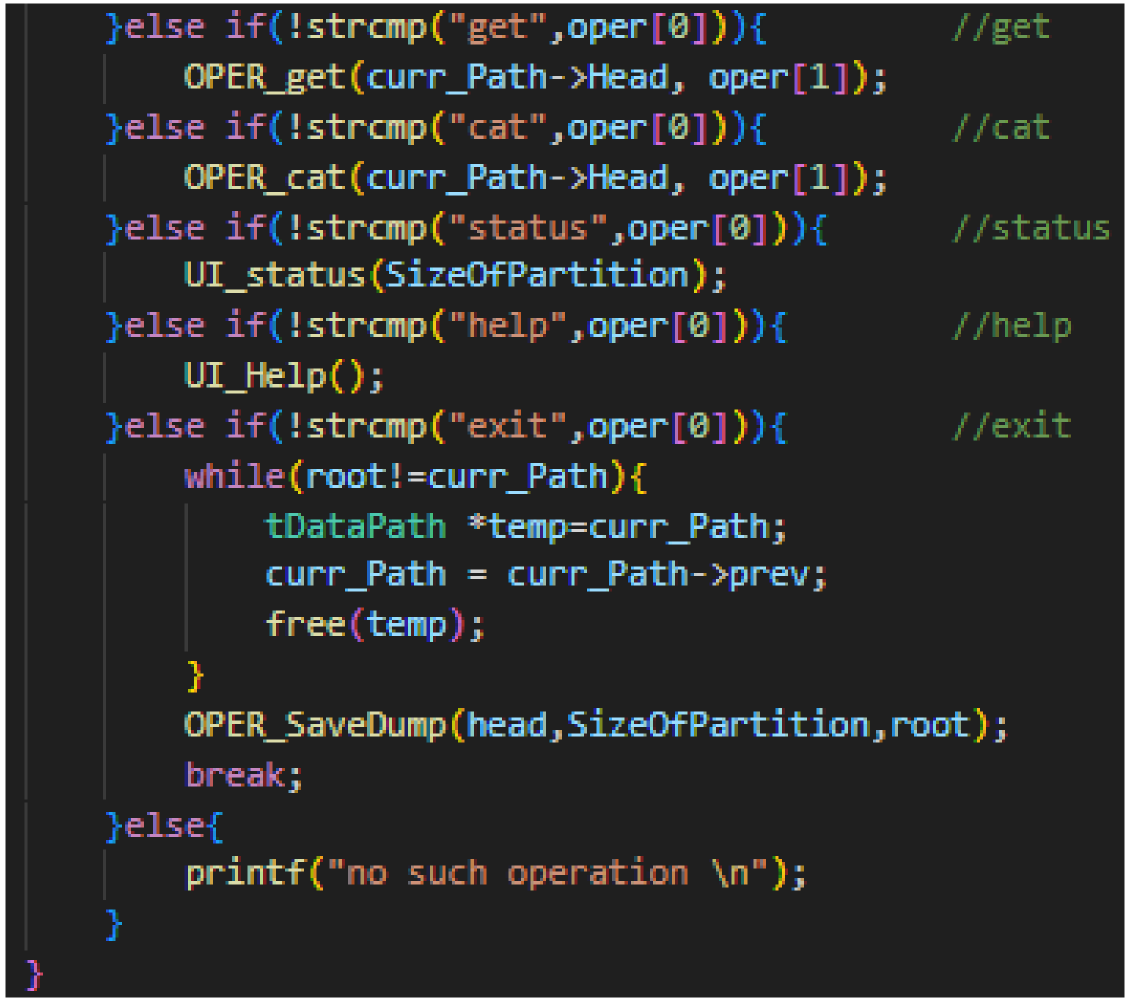


```
int SizeOfPartition=-1;              //新建之空間大小
char oper[2][10];                    //輸入轉換為操作及目的檔案名稱
tDataTree *load;

//--------------------------------------------------------------

load=UI_SelectFunc_Init(&SizeOfPartition);        //空間初始化
UI_Help();

//--------------------------------------------------------------
tDataHead *head=Create_Init_DataHead("root");    //資料節點管理
tDataPath *root=Create_Init_DataPath(head);      //路徑管理
tDataPath *curr_Path=root;                       //當前路徑

//--------------------------------------------------------------

if(load!=NULL){
    head->next=load;
}
```

main

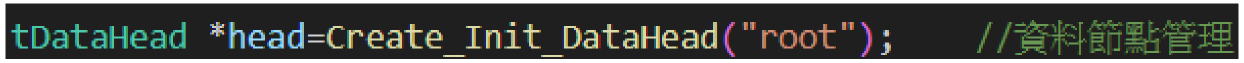


```c
while(1){
    strcpy(oper[0],"\0");                    //清空
    strcpy(oper[1],"\0");
    UI_SelectFunc_Oper(oper,root,curr_Path); //選擇操作

    if(!strcmp("ls",oper[0])){               //ls
        OPER_ls(curr_Path->Head);
    }else if(!strcmp("cd",oper[0])){
        int FoR;                             //切換路徑至上層或下層
        FoR=OPER_cd(oper[1],root,curr_Path);
        if(FoR==1){
            curr_Path = curr_Path->next;     //下層=>路徑往下走
        }else if(FoR==0){
            tDataPath *temp=curr_Path;       //上層=>路徑往上走
            curr_Path = curr_Path->prev;
            free(temp);                      //並釋放掉DataPath空間
        }
    }else if(!strcmp("rm",oper[0])){         //rm
        OPER_rm(curr_Path->Head,oper[1]);
    }else if(!strcmp("mkdir",oper[0])){ (int)1 mkdir
        OPER_mkdir(curr_Path->Head,oper[1]);
    }else if(!strcmp("rmdir",oper[0])){      //rmdir
        OPER_rmdir(curr_Path->Head,oper[1]);
    }else if(!strcmp("put",oper[0])){        //put
        OPER_put(curr_Path->Head,oper[1]);
```

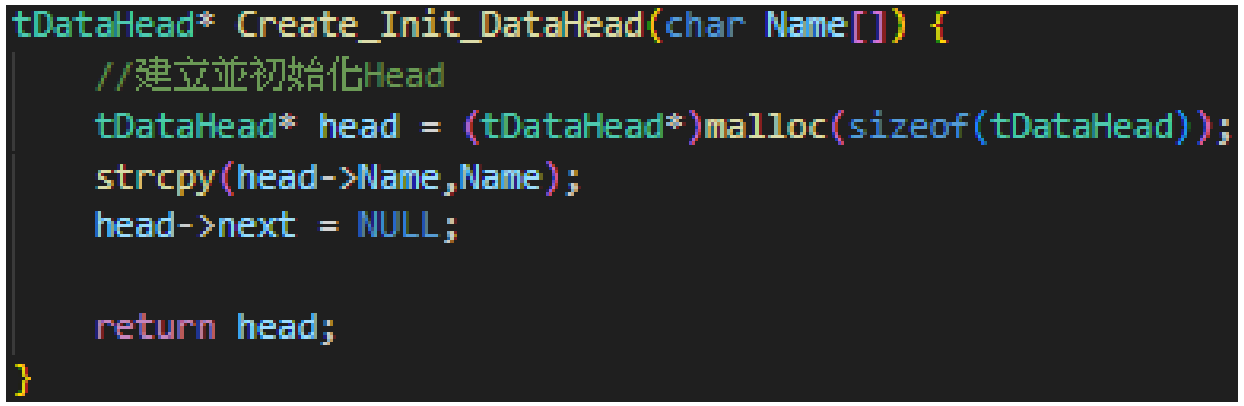


```c
    }else if(!strcmp("get",oper[0])){        //get
        OPER_get(curr_Path->Head, oper[1]);
    }else if(!strcmp("cat",oper[0])){        //cat
        OPER_cat(curr_Path->Head, oper[1]);
    }else if(!strcmp("status",oper[0])){     //status
        UI_status(SizeOfPartition);
    }else if(!strcmp("help",oper[0])){       //help
        UI_Help();
    }else if(!strcmp("exit",oper[0])){       //exit
        while(root!=curr_Path){
            tDataPath *temp=curr_Path;
            curr_Path = curr_Path->prev;
            free(temp);
        }
        OPER_SaveDump(head,SizeOfPartition,root);
        break;
    }else{
        printf("no such operation \n");
    }
}
```

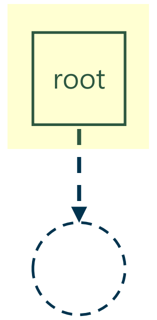
Creat_Init_DataHead & Creat_Init_DataPath

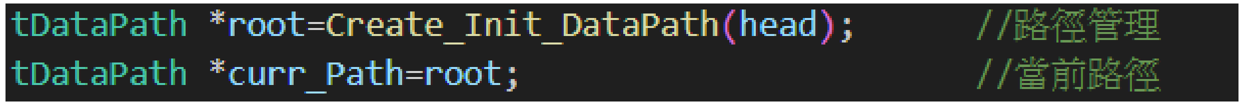


```c
tDataHead *head=Create_Init_DataHead("root");     //資料節點管理
```

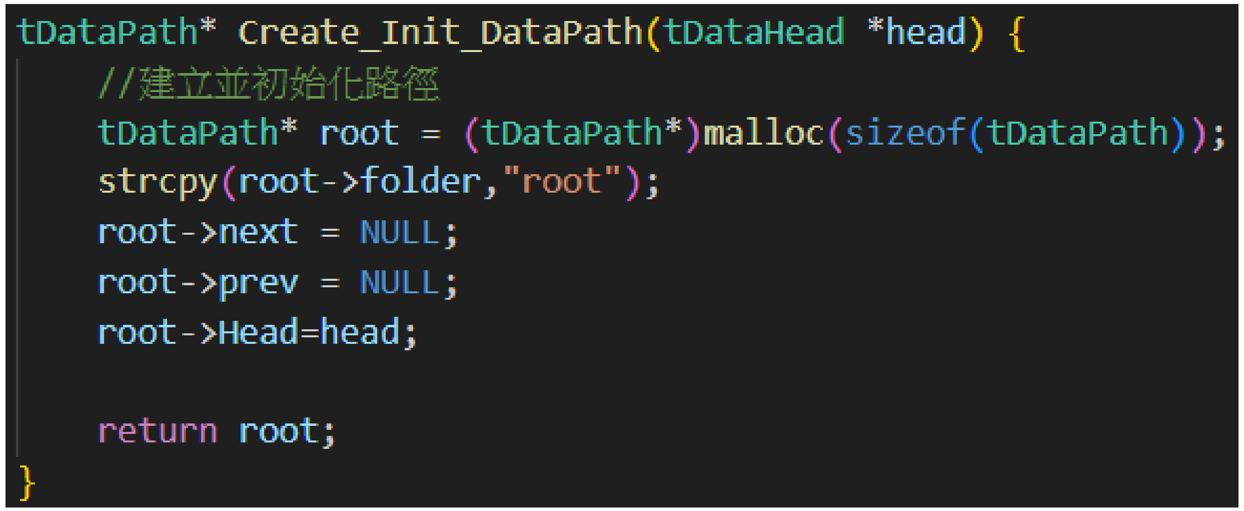


```c
tDataHead* Create_Init_DataHead(char Name[]) {
    //建立並初始化Head
    tDataHead* head = (tDataHead*)malloc(sizeof(tDataHead));
    strcpy(head->Name,Name);
    head->next = NULL;

    return head;
}
```

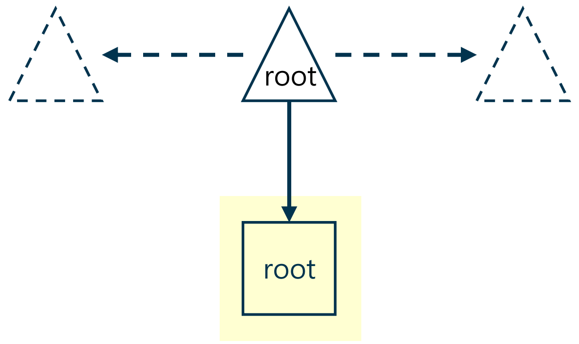




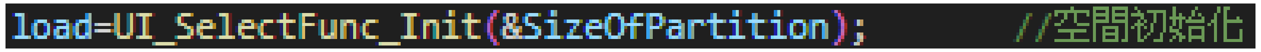
```c
tDataPath *root=Create_Init_DataPath(head);       //路徑管理
tDataPath *curr_Path=root;                        //當前路徑
```



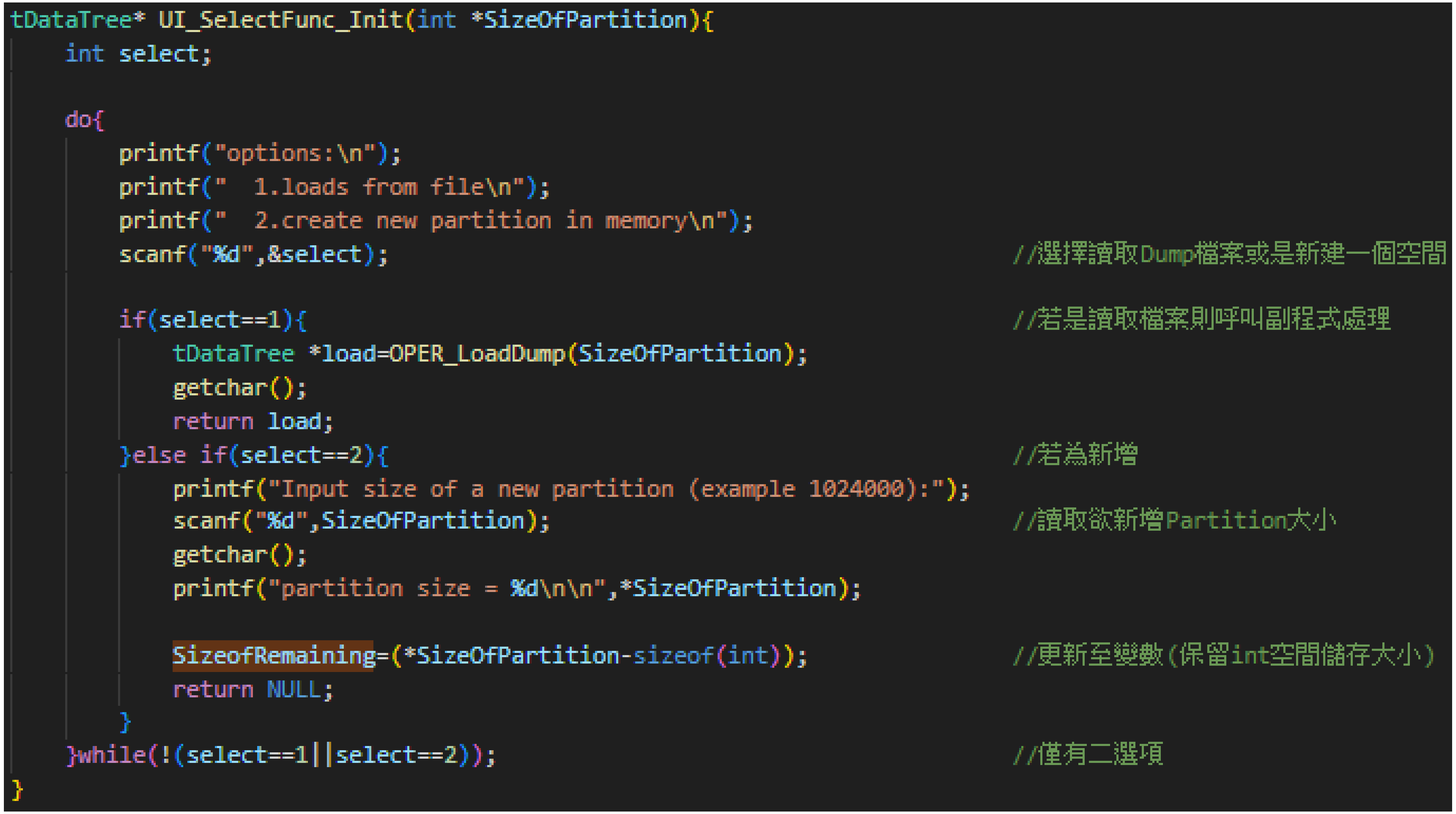
```c
tDataPath* Create_Init_DataPath(tDataHead *head) {
    //建立並初始化路徑
    tDataPath* root = (tDataPath*)malloc(sizeof(tDataPath));
    strcpy(root->folder,"root");
    root->next = NULL;
    root->prev = NULL;
    root->Head=head;

    return root;
}
```



UI_SelectFunc_Init

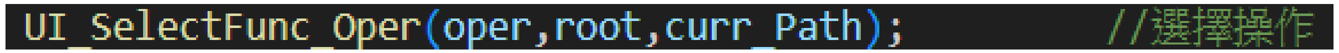


```c
load=UI_SelectFunc_Init(&SizeOfPartition);          //空間初始化
```

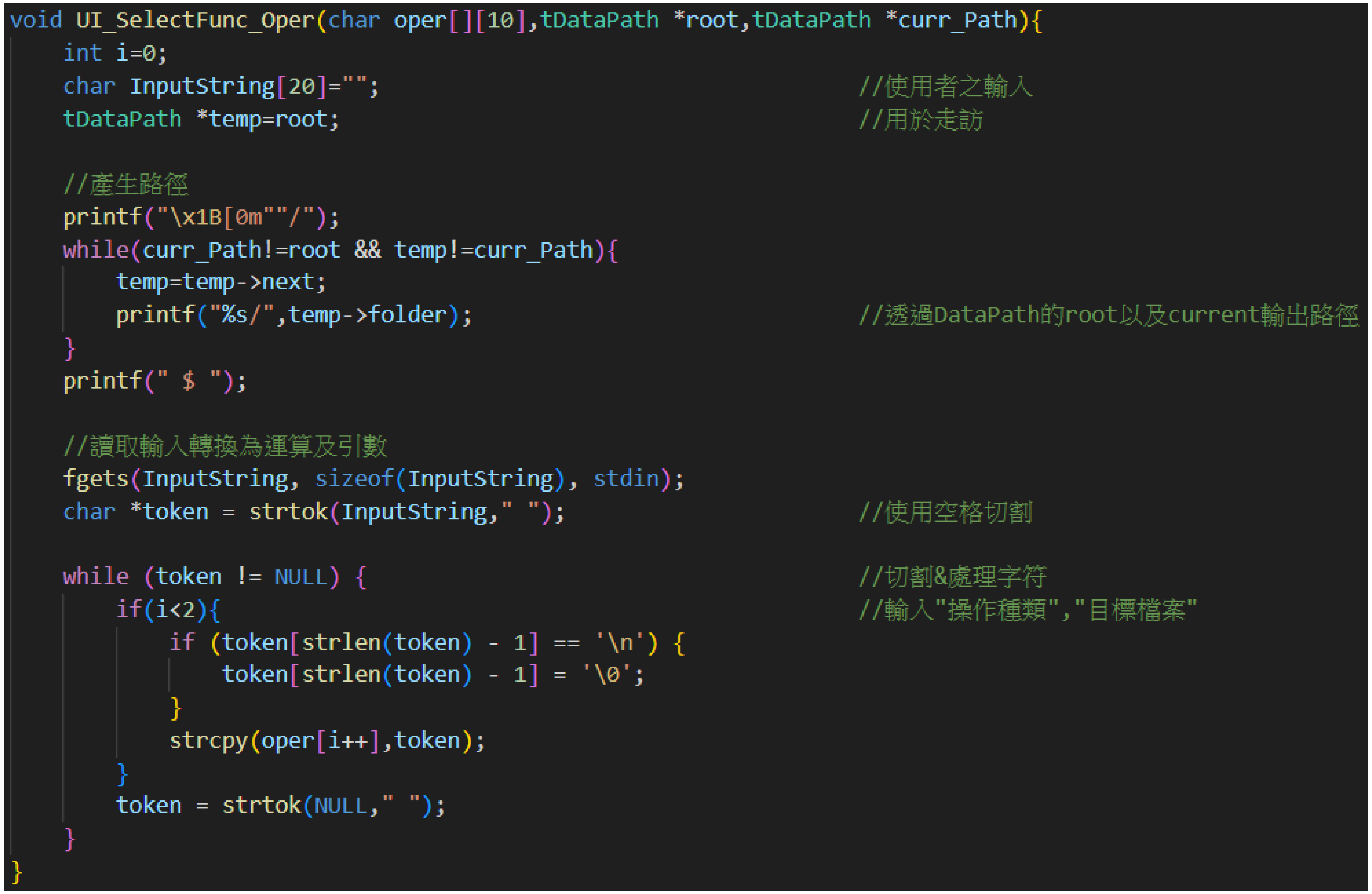


```c
tDataTree* UI_SelectFunc_Init(int *SizeOfPartition){
    int select;

    do{
        printf("options:\n");
        printf("  1.loads from file\n");
        printf("  2.create new partition in memory\n");
        scanf("%d",&select);                                          //選擇讀取Dump檔案或是新建一個空間

        if(select==1){                                                //若是讀取檔案則呼叫副程式處理
            tDataTree *load=OPER_LoadDump(SizeOfPartition);
            getchar();
            return load;
        }else if(select==2){                                          //若為新增
            printf("Input size of a new partition (example 1024000):");
            scanf("%d",SizeOfPartition);                              //讀取欲新增Partition大小
            getchar();
            printf("partition size = %d\n\n",*SizeOfPartition);

            SizeofRemaining=(*SizeOfPartition-sizeof(int));           //更新至變數(保留int空間儲存大小)
            return NULL;
        }
    }while(!(select==1||select==2));                                  //僅有二選項
}
```

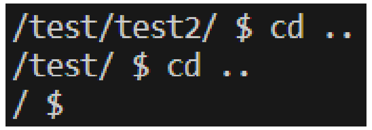
UI_SelectFunc_Oper



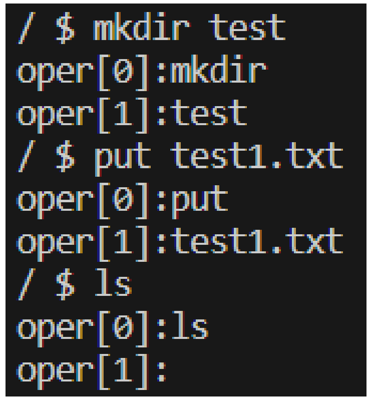
```
UI_SelectFunc_Oper(oper,root,curr_Path);          //選擇操作
```



```
void UI_SelectFunc_Oper(char oper[][10],tDataPath *root,tDataPath *curr_Path){
    int i=0;
    char InputString[20]="";                        //使用者之輸入
    tDataPath *temp=root;                           //用於走訪

    //產生路徑
    printf("\x1B[0m""/");
    while(curr_Path!=root && temp!=curr_Path){
        temp=temp->next;
        printf("%s/",temp->folder);                 //透過DataPath的root以及current輸出路徑
    }
    printf(" $ ");

    //讀取輸入轉換為運算及引數
    fgets(InputString, sizeof(InputString), stdin);
    char *token = strtok(InputString," ");          //使用空格切割

    while (token != NULL) {                         //切割&處理字符
        if(i<2){                                    //輸入"操作種類","目標檔案"
            if (token[strlen(token) - 1] == '\n') {
                token[strlen(token) - 1] = '\0';
            }
            strcpy(oper[i++],token);
        }
        token = strtok(NULL," ");
    }
}
```

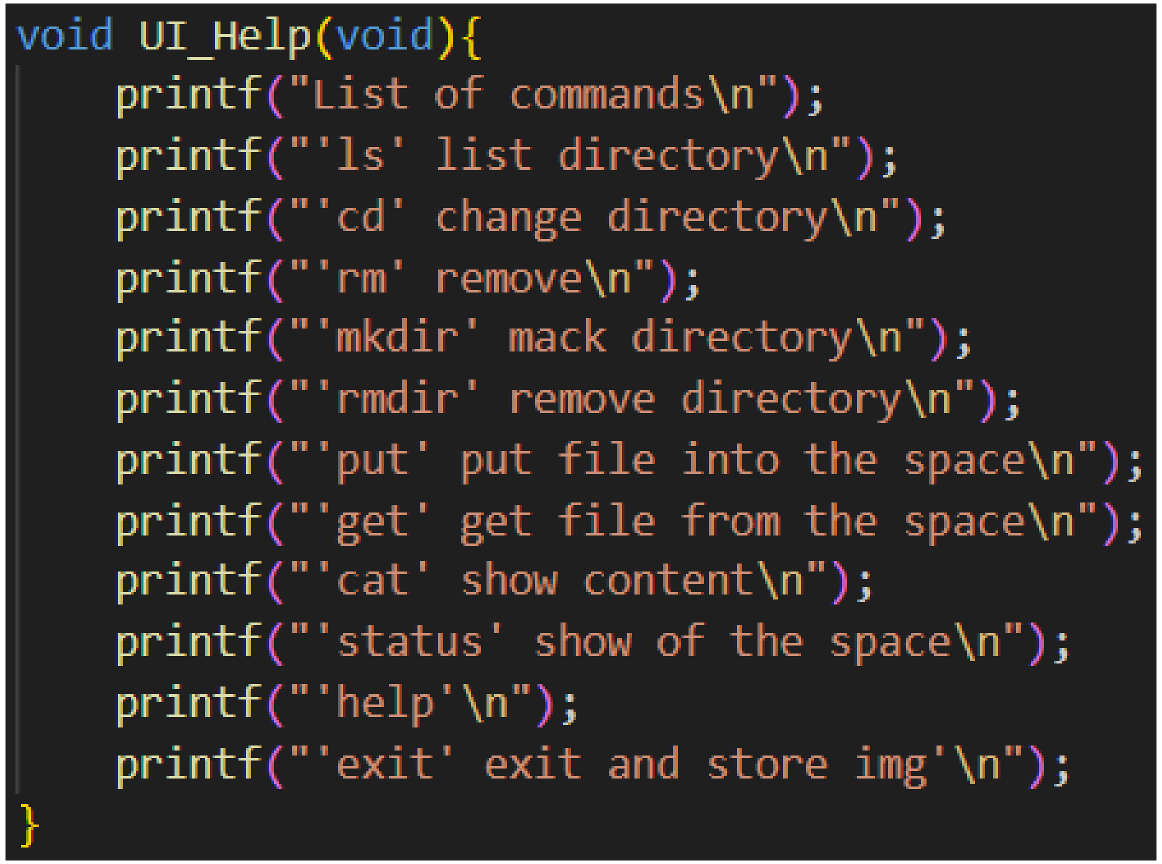


```
/test/test2/ $ cd ..
/test/ $ cd ..
/ $
```

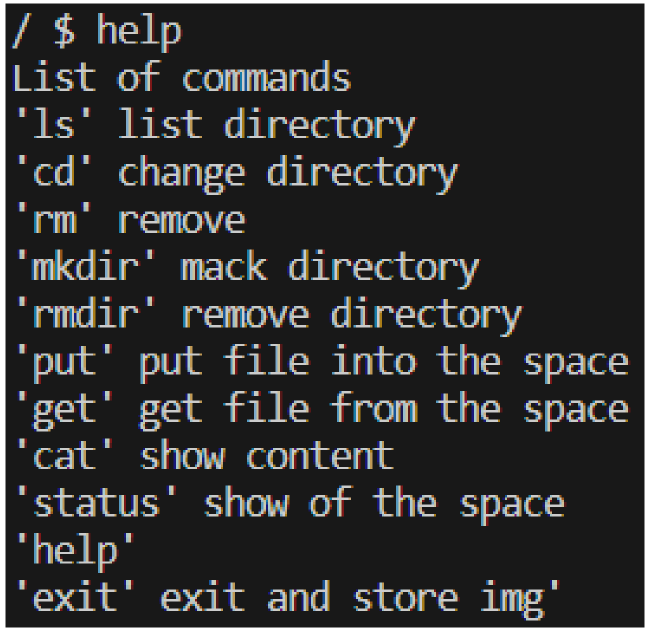


```
/ $ mkdir test
oper[0]:mkdir
oper[1]:test
/ $ put test1.txt
oper[0]:put
oper[1]:test1.txt
/ $ ls
oper[0]:ls
oper[1]:
```

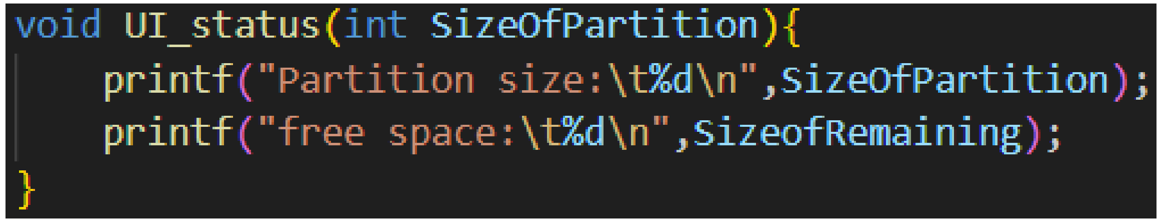
UI



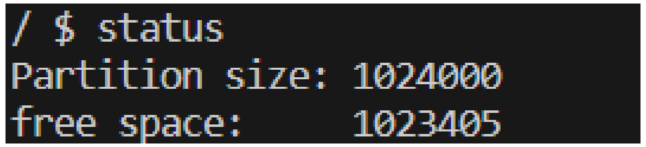
```c
void UI_Help(void){
    printf("List of commands\n");
    printf("'ls' list directory\n");
    printf("'cd' change directory\n");
    printf("'rm' remove\n");
    printf("'mkdir' mack directory\n");
    printf("'rmdir' remove directory\n");
    printf("'put' put file into the space\n");
    printf("'get' get file from the space\n");
    printf("'cat' show content\n");
    printf("'status' show of the space\n");
    printf("'help'\n");
    printf("'exit' exit and store img'\n");
}
```



```
/ $ help
List of commands
'ls' list directory
'cd' change directory
'rm' remove
'mkdir' mack directory
'rmdir' remove directory
'put' put file into the space
'get' get file from the space
'cat' show content
'status' show of the space
'help'
'exit' exit and store img'
```

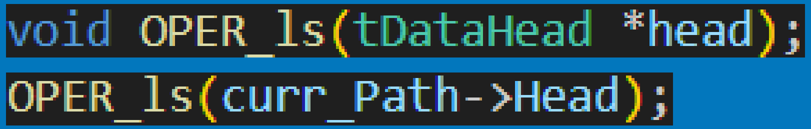


```c
void UI_status(int SizeOfPartition){
    printf("Partition size:\t%d\n",SizeOfPartition);
    printf("free space:\t%d\n",SizeofRemaining);
}
```

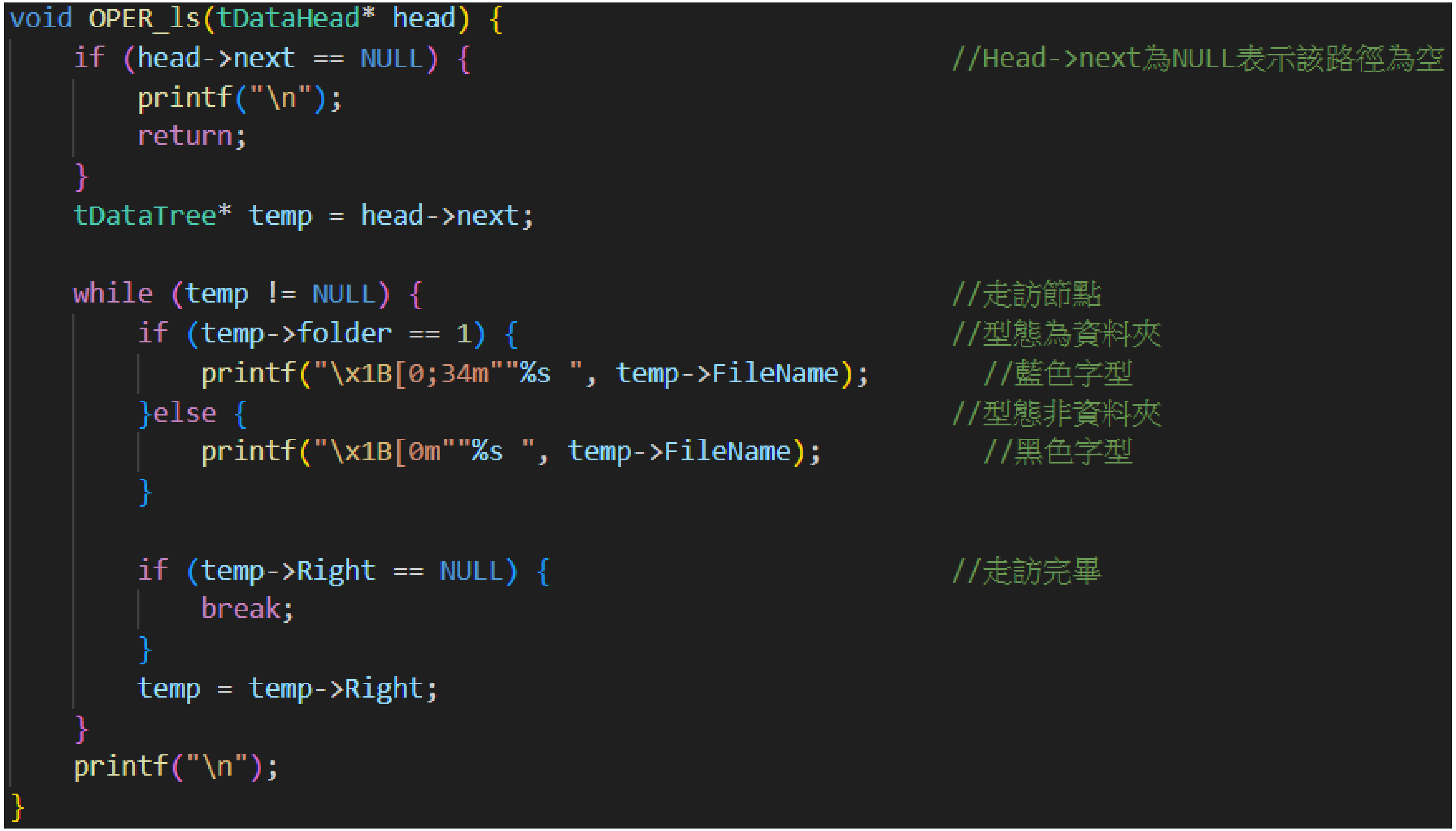


```
/ $ status
Partition size: 1024000
free space:     1023405
```

OPER_ls

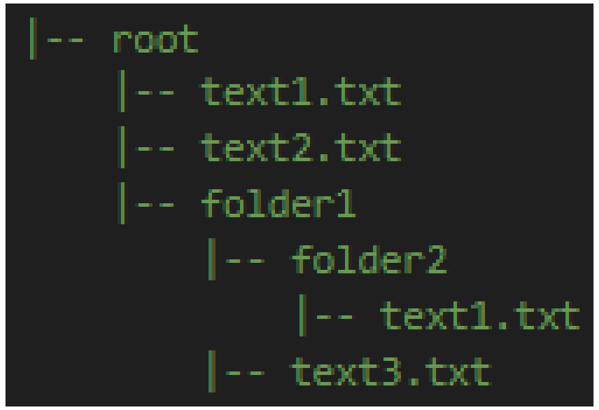


```
void OPER_ls(tDataHead *head);
OPER_ls(curr_Path->Head);
```



```
void OPER_ls(tDataHead* head) {
    if (head->next == NULL) {                                   //Head->next為NULL表示該路徑為空
        printf("\n");
        return;
    }
    tDataTree* temp = head->next;

    while (temp != NULL) {                                      //走訪節點
        if (temp->folder == 1) {                                //型態為資料夾
            printf("\x1B[0;34m""%s ", temp->FileName);            //藍色字型
        }else {                                                 //型態非資料夾
            printf("\x1B[0m""%s ", temp->FileName);               //黑色字型
        }

        if (temp->Right == NULL) {                              //走訪完畢
            break;
        }
        temp = temp->Right;
    }
    printf("\n");
}
```

範例



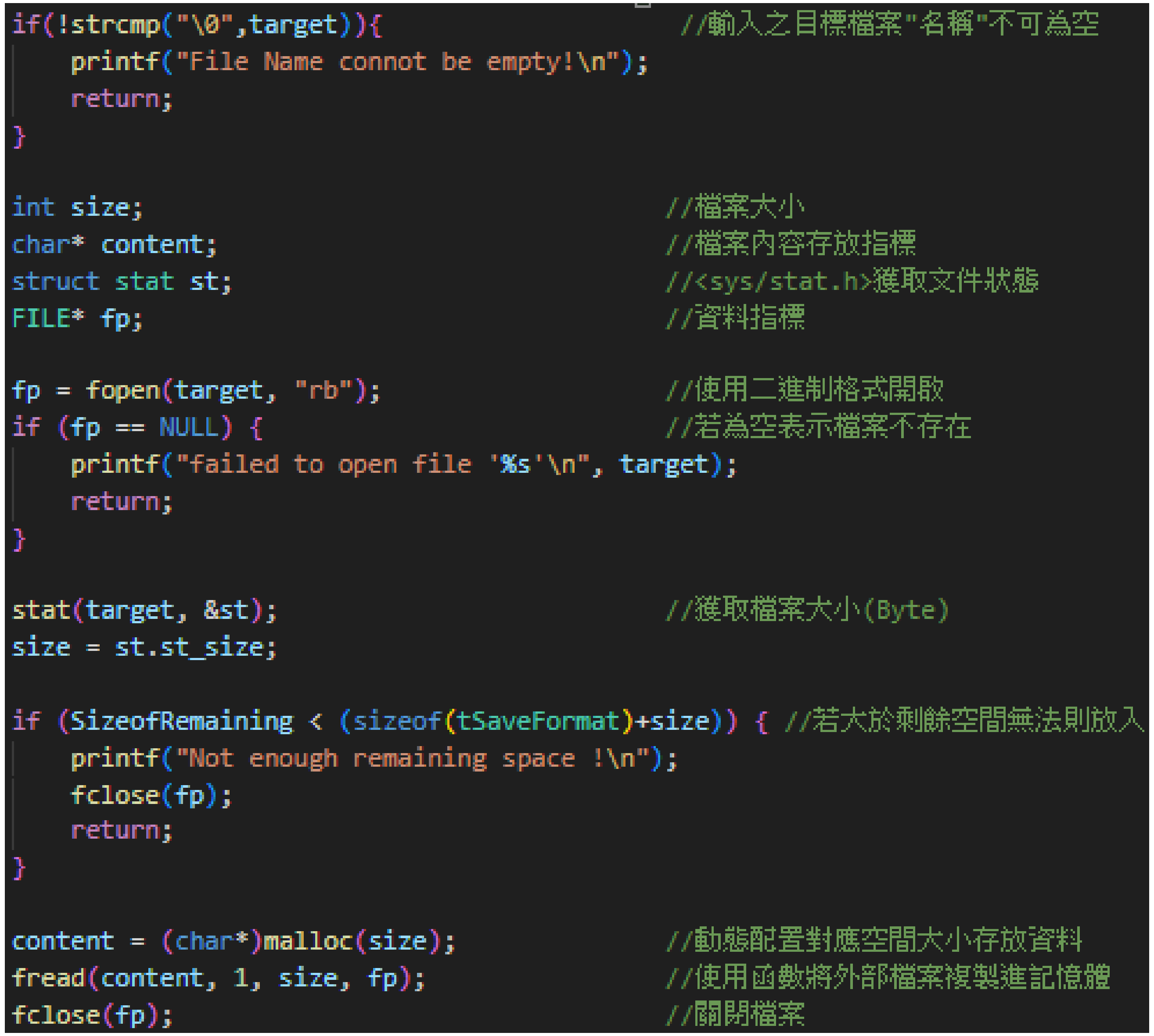
```
|-- root
    |-- text1.txt
    |-- text2.txt
    |-- folder1
        |-- folder2
            |-- text1.txt
        |-- text3.txt
```

以下將以建立左圖之資料路徑之順序介紹其餘副程式

OPER_put

```
void OPER_put(tDataHead *head,char target[]);
OPER_put(curr_Path->Head,oper[1]);
```

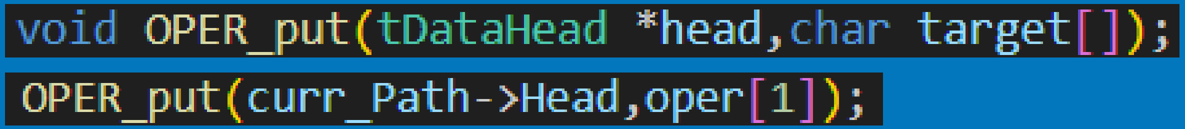
```
/ $ put text1.txt
```



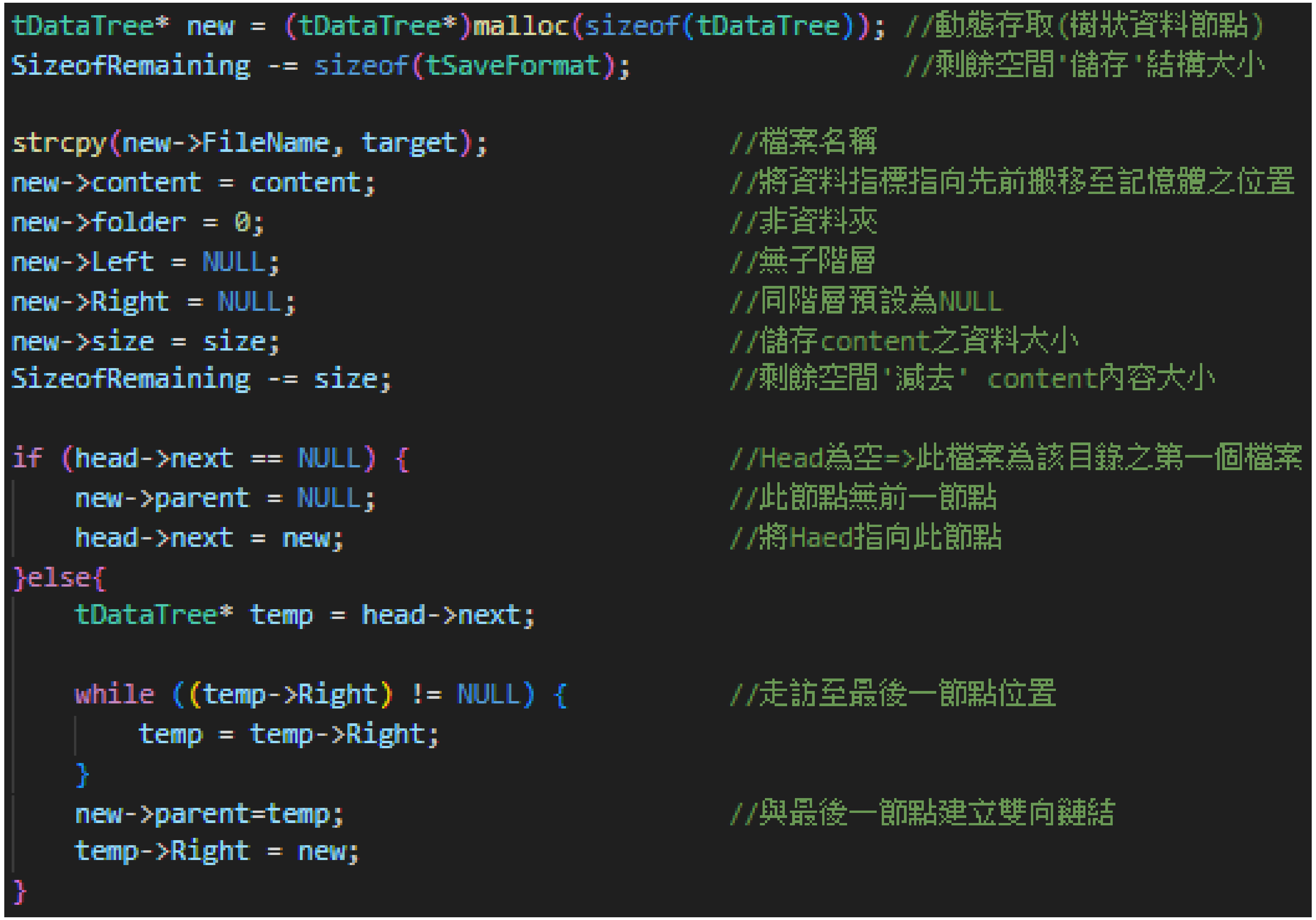
```
if(!strcmp("\0",target)){                         //輸入之目標檔案"名稱"不可為空
    printf("File Name connot be empty!\n");
    return;
}

int size;                                        //檔案大小
char* content;                                   //檔案內容存放指標
struct stat st;                                  //<sys/stat.h>獲取文件狀態
FILE* fp;                                        //資料指標

fp = fopen(target, "rb");                        //使用二進制格式開啟
if (fp == NULL) {                                //若為空表示檔案不存在
    printf("failed to open file '%s'\n", target);
    return;
}

stat(target, &st);                               //獲取檔案大小(Byte)
size = st.st_size;

if (SizeofRemaining < (sizeof(tSaveFormat)+size)) { //若大於剩餘空間無法則放入
    printf("Not enough remaining space !\n");
    fclose(fp);
    return;
}

content = (char*)malloc(size);                   //動態配置對應空間大小存放資料
fread(content, 1, size, fp);                     //使用函數將外部檔案複製進記憶體
fclose(fp);                                      //關閉檔案
```

資料讀取以及搬移

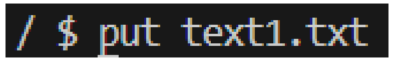
OPER_put



```c
void OPER_put(tDataHead *head,char target[]);
OPER_put(curr_Path->Head,oper[1]);
```



```c
tDataTree* new = (tDataTree*)malloc(sizeof(tDataTree)); //動態存取(樹狀資料節點)
SizeofRemaining -= sizeof(tSaveFormat);                 //剩餘空間'儲存'結構大小

strcpy(new->FileName, target);          //檔案名稱
new->content = content;                 //將資料指標指向先前搬移至記憶體之位置
new->folder = 0;                        //非資料夾
new->Left = NULL;                       //無子階層
new->Right = NULL;                      //同階層預設為NULL
new->size = size;                       //儲存content之資料大小
SizeofRemaining -= size;                //剩餘空間'減去' content內容大小

if (head->next == NULL) {               //Head為空=>此檔案為該目錄之第一個檔案
    new->parent = NULL;                 //此節點無前一節點
    head->next = new;                   //將Haed指向此節點
}else{
    tDataTree* temp = head->next;

    while ((temp->Right) != NULL) {     //走訪至最後一節點位置
        temp = temp->Right;
    }
    new->parent=temp;                   //與最後一節點建立雙向鏈結
    temp->Right = new;
}
```



```
/ $ put text1.txt
```

DataTree結構維護

OPER_put

```
void OPER_put(tDataHead *head,char target[]);
OPER_put(curr_Path->Head,oper[1]);
```

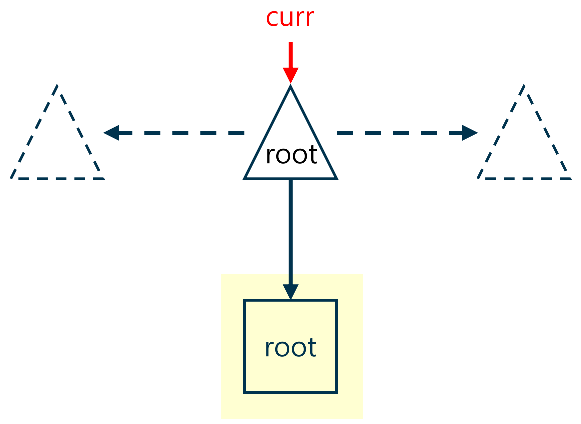
```
/ $ put text1.txt
```

```
/ $ ls
text1.txt
```

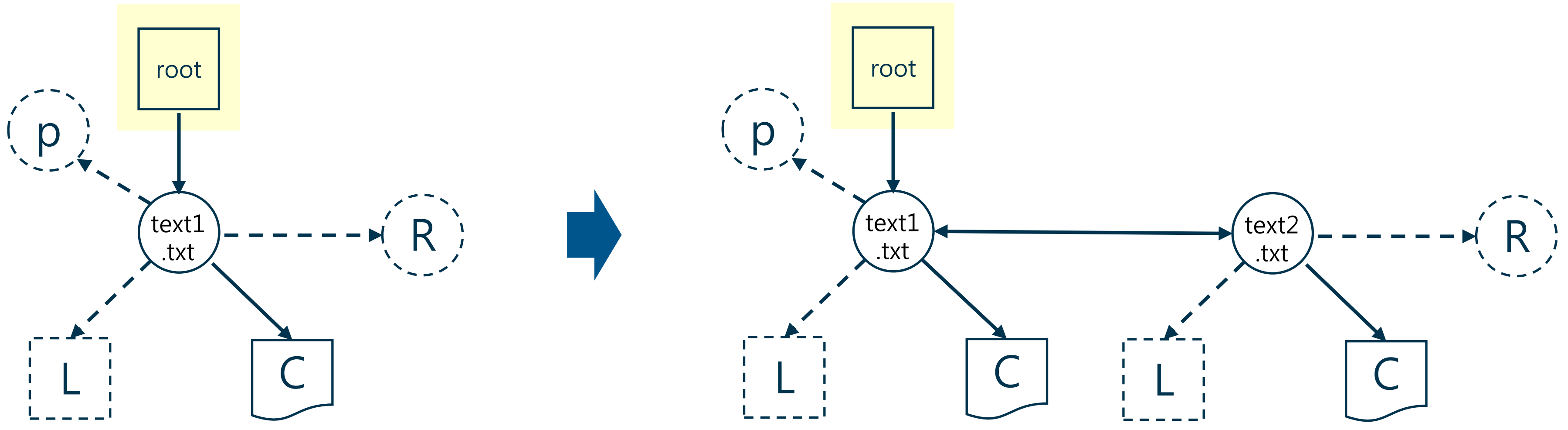
```
/ $ put text2.txt
```

```
/ $ ls
text1.txt text2.txt
```

DataPath :

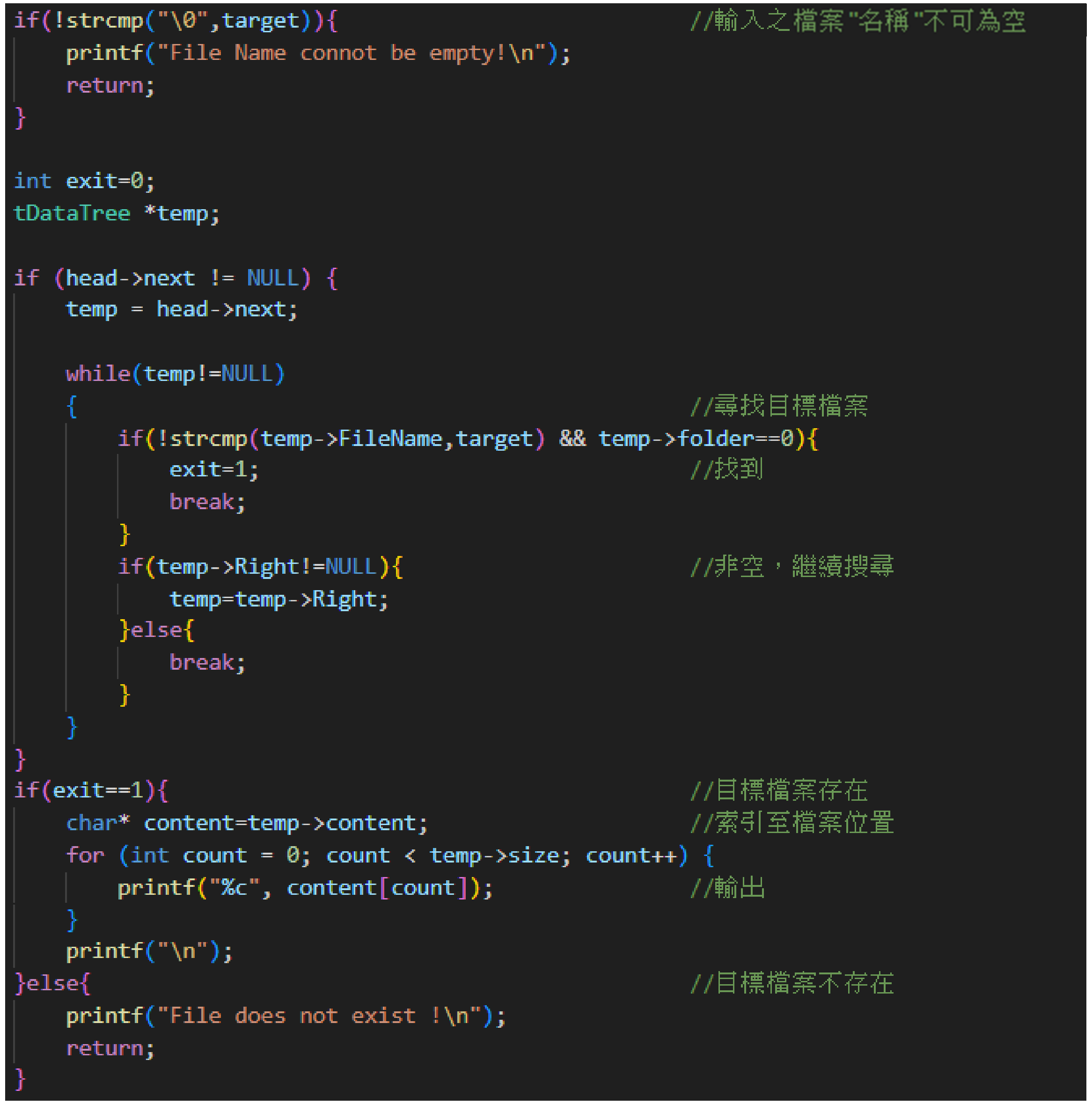


DataTree :

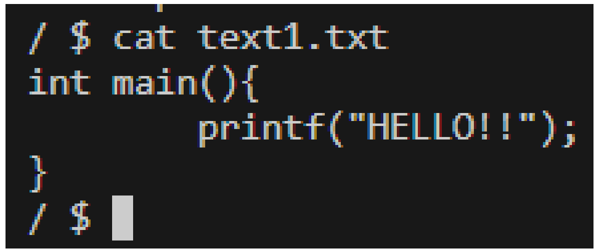


OPER_cat

```
void OPER_cat(tDataHead *head,char target[]);
OPER_cat(curr_Path->Head, oper[1]);
```



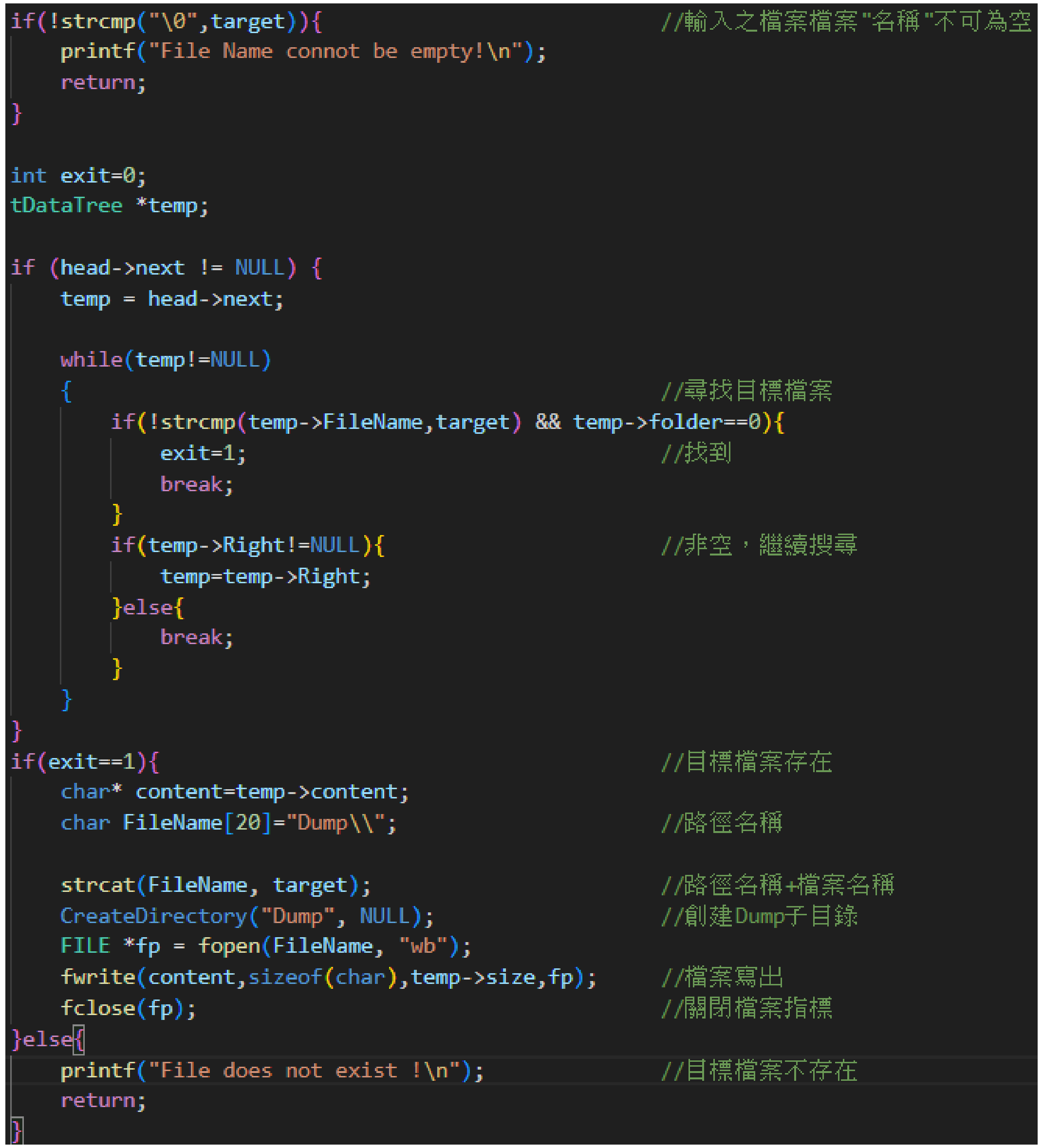
```
if(!strcmp("\0",target)){                               //輸入之檔案"名稱"不可為空
    printf("File Name connot be empty!\n");
    return;
}

int exit=0;
tDataTree *temp;

if (head->next != NULL) {
    temp = head->next;

    while(temp!=NULL)
    {                                                   //尋找目標檔案
        if(!strcmp(temp->FileName,target) && temp->folder==0){
            exit=1;                                     //找到
            break;
        }
        if(temp->Right!=NULL){                          //非空，繼續搜尋
            temp=temp->Right;
        }else{
            break;
        }
    }
}
if(exit==1){                                            //目標檔案存在
    char* content=temp->content;                        //索引至檔案位置
    for (int count = 0; count < temp->size; count++) {
        printf("%c", content[count]);                   //輸出
    }
    printf("\n");
}else{                                                  //目標檔案不存在
    printf("File does not exist !\n");
    return;
}
```



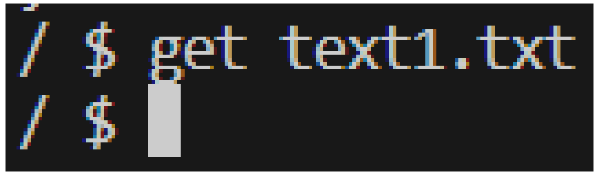
```
/ $ cat text1.txt
int main(){
        printf("HELLO!!");
}
/ $ 
```

OPER_get

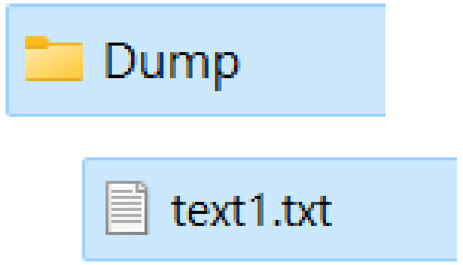
```
OPER_get(curr_Path->Head, oper[1]);
void OPER_get(tDataHead *head,char target[]);
```



```
if(!strcmp("\0",target)){                               //輸入之檔案檔案"名稱"不可為空
    printf("File Name connot be empty!\n");
    return;
}

int exit=0;
tDataTree *temp;

if (head->next != NULL) {
    temp = head->next;

    while(temp!=NULL)
    {                                                   //尋找目標檔案
        if(!strcmp(temp->FileName,target) && temp->folder==0){
            exit=1;                                     //找到
            break;
        }
        if(temp->Right!=NULL){                          //非空，繼續搜尋
            temp=temp->Right;
        }else{
            break;
        }
    }
}
if(exit==1){                                            //目標檔案存在
    char* content=temp->content;
    char FileName[20]="Dump\\";                         //路徑名稱

    strcat(FileName, target);                           //路徑名稱+檔案名稱
    CreateDirectory("Dump", NULL);                      //創建Dump子目錄
    FILE *fp = fopen(FileName, "wb");
    fwrite(content,sizeof(char),temp->size,fp);         //檔案寫出
    fclose(fp);                                         //關閉檔案指標
}else{
    printf("File does not exist !\n");                  //目標檔案不存在
    return;
}
```

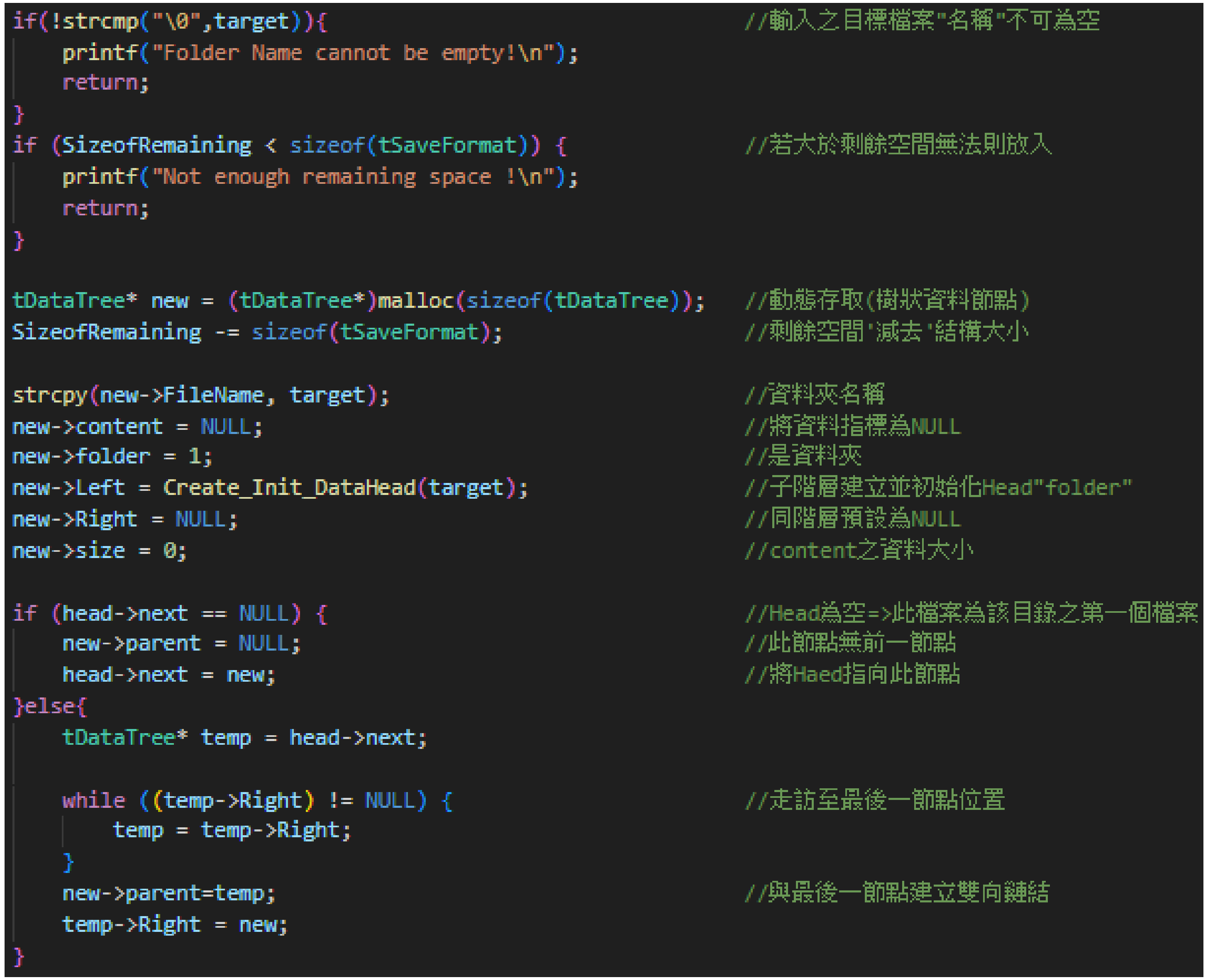


```
/ $ get text1.txt
/ $ 
```



OPER_mkdir

```c
void OPER_mkdir(tDataHead *head,char target[]);
OPER_mkdir(curr_Path->Head,oper[1]);
```

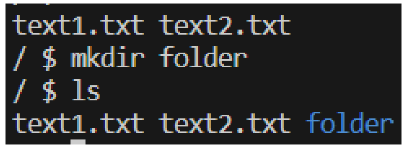


```c
if(!strcmp("\0",target)){                               //輸入之目標檔案"名稱"不可為空
    printf("Folder Name cannot be empty!\n");
    return;
}
if (SizeofRemaining < sizeof(tSaveFormat)) {           //若大於剩餘空間無法則放入
    printf("Not enough remaining space !\n");
    return;
}

tDataTree* new = (tDataTree*)malloc(sizeof(tDataTree));  //動態存取(樹狀資料節點)
SizeofRemaining -= sizeof(tSaveFormat);                 //剩餘空間'減去'結構大小

strcpy(new->FileName, target);                          //資料夾名稱
new->content = NULL;                                    //將資料指標為NULL
new->folder = 1;                                        //是資料夾
new->Left = Create_Init_DataHead(target);               //子階層建立並初始化Head"folder"
new->Right = NULL;                                      //同階層預設為NULL
new->size = 0;                                          //content之資料大小

if (head->next == NULL) {                               //Head為空=>此檔案為該目錄之第一個檔案
    new->parent = NULL;                                 //此節點無前一節點
    head->next = new;                                   //將Haed指向此節點
}else{
    tDataTree* temp = head->next;

    while ((temp->Right) != NULL) {                     //走訪至最後一節點位置
        temp = temp->Right;
    }
    new->parent=temp;                                   //與最後一節點建立雙向鏈結
    temp->Right = new;
}
```

插入資料夾之樹狀節點

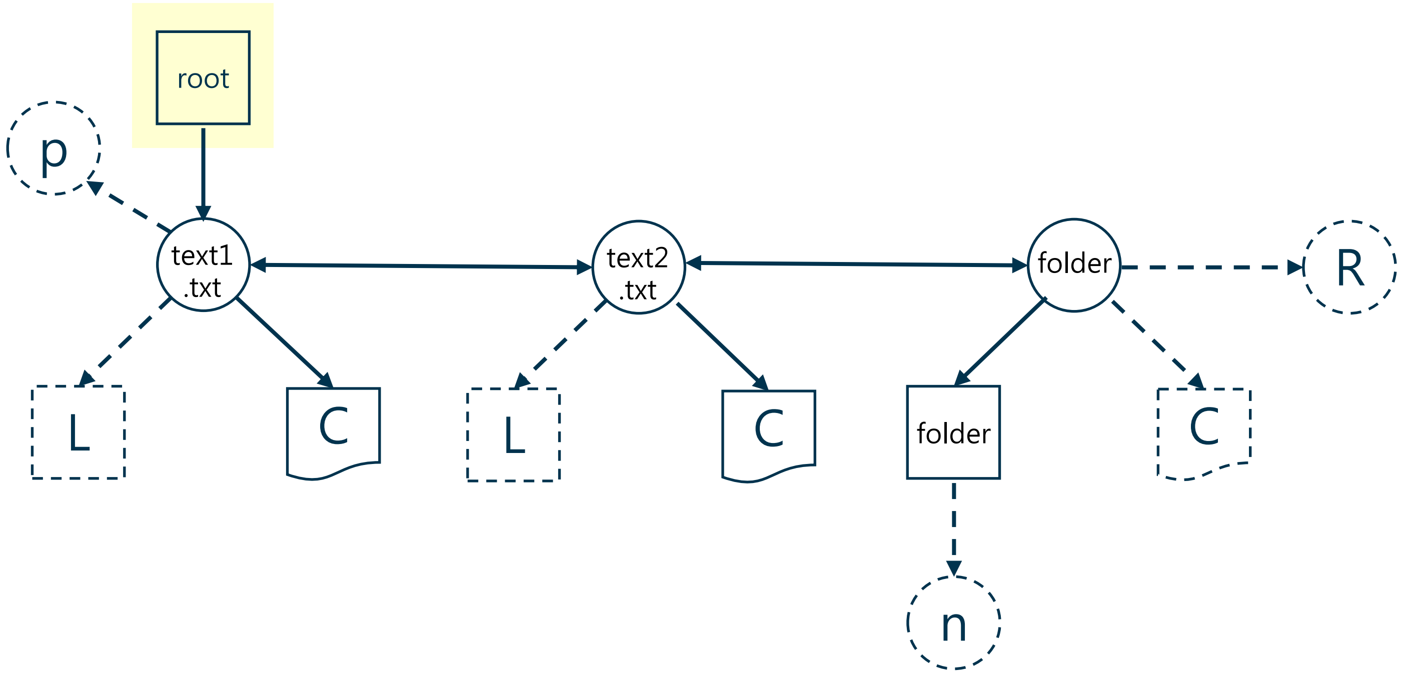
OPER_mkdir

```
void OPER_mkdir(tDataHead *head,char target[]);
OPER_mkdir(curr_Path->Head,oper[1]);
```



```
text1.txt text2.txt
/ $ mkdir folder
/ $ ls
text1.txt text2.txt folder
```

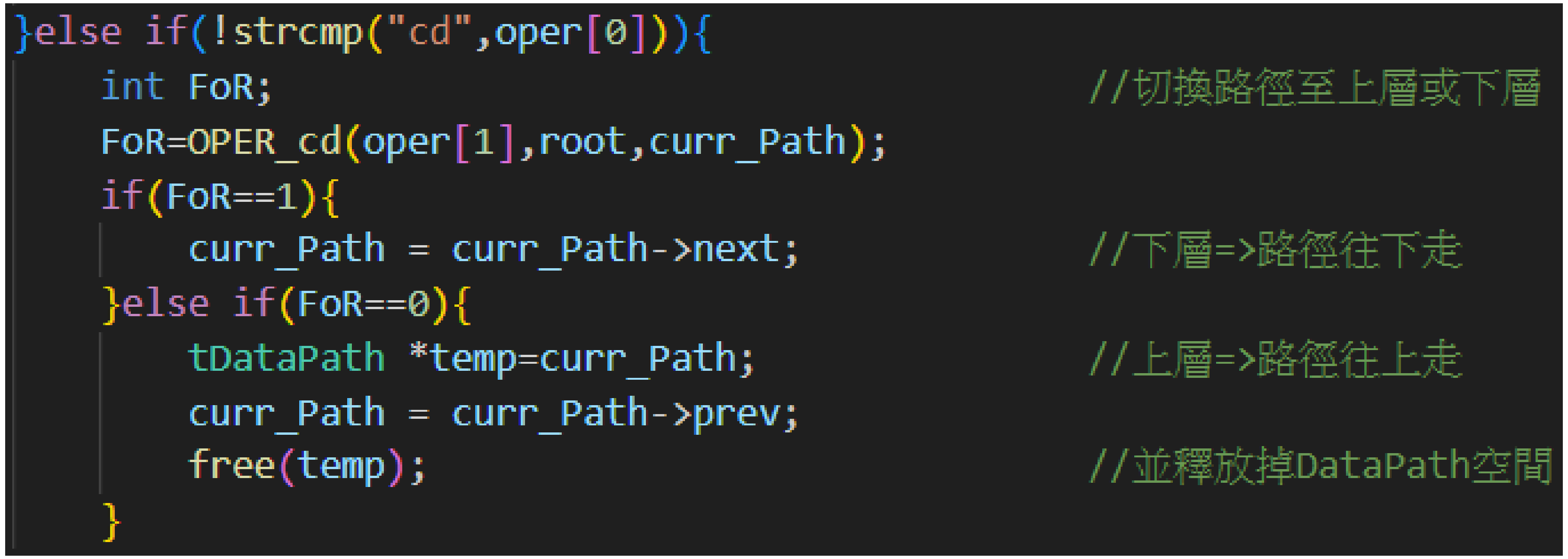
DataTree :



OPER_cd

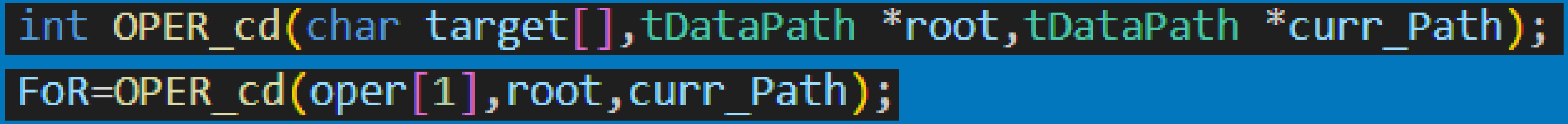
```c
int OPER_cd(char target[],tDataPath *root,tDataPath *curr_Path);
```

```c
FoR=OPER_cd(oper[1],root,curr_Path);
```

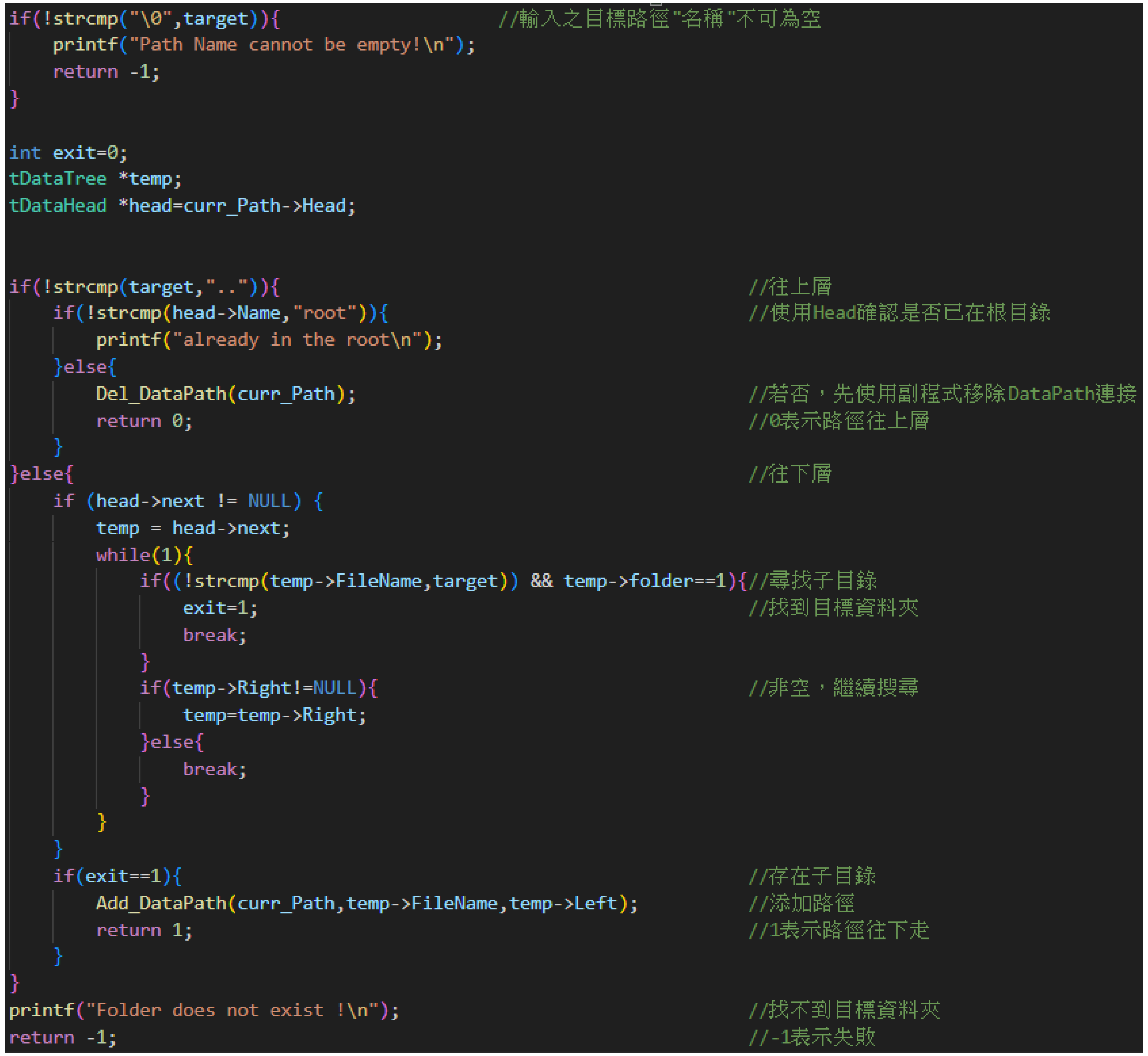


```c
}else if(!strcmp("cd",oper[0])){
    int FoR;                                                //切換路徑至上層或下層
    FoR=OPER_cd(oper[1],root,curr_Path);
    if(FoR==1){
        curr_Path = curr_Path->next;                        //下層=>路徑往下走
    }else if(FoR==0){
        tDataPath *temp=curr_Path;                          //上層=>路徑往上走
        curr_Path = curr_Path->prev;
        free(temp);                                         //並釋放掉DataPath空間
    }
```

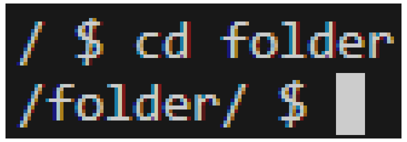
OPER_cd



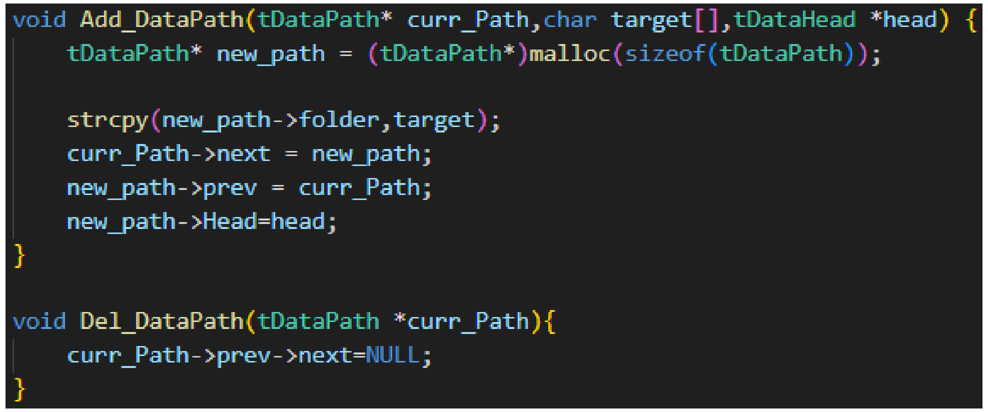
```c
int OPER_cd(char target[],tDataPath *root,tDataPath *curr_Path);
FoR=OPER_cd(oper[1],root,curr_Path);
```



```c
if(!strcmp("\0",target)){                    //輸入之目標路徑"名稱"不可為空
    printf("Path Name cannot be empty!\n");
    return -1;
}

int exit=0;
tDataTree *temp;
tDataHead *head=curr_Path->Head;


if(!strcmp(target,"..")){                                   //往上層
    if(!strcmp(head->Name,"root")){                         //使用Head確認是否已在根目錄
        printf("already in the root\n");
    }else{
        Del_DataPath(curr_Path);                            //若否，先使用副程式移除DataPath連接
        return 0;                                           //0表示路徑往上層
    }
}else{                                                      //往下層
    if (head->next != NULL) {
        temp = head->next;
        while(1){
            if((!strcmp(temp->FileName,target)) && temp->folder==1){//尋找子目錄
                exit=1;                                     //找到目標資料夾
                break;
            }
            if(temp->Right!=NULL){                          //非空，繼續搜尋
                temp=temp->Right;
            }else{
                break;
            }
        }
    }
    if(exit==1){                                            //存在子目錄
        Add_DataPath(curr_Path,temp->FileName,temp->Left);  //添加路徑
        return 1;                                           //1表示路徑往下走
    }
}
printf("Folder does not exist !\n");                        //找不到目標資料夾
return -1;                                                  //-1表示失敗
```



```
/ $ cd folder
/folder/ $ 
```



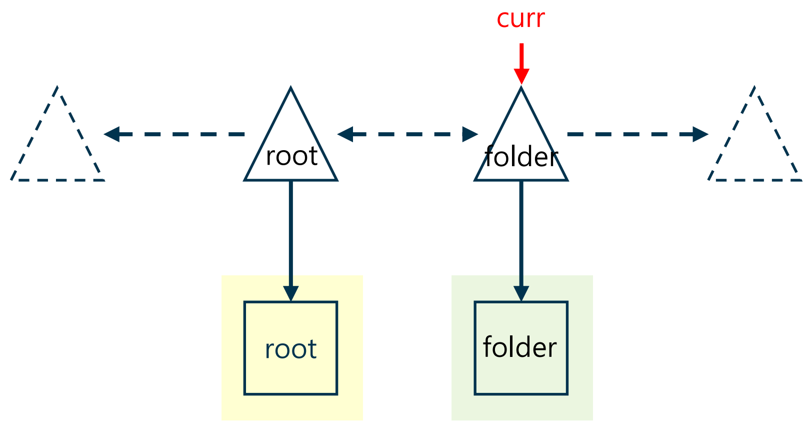
```c
void Add_DataPath(tDataPath* curr_Path,char target[],tDataHead *head) {
    tDataPath* new_path = (tDataPath*)malloc(sizeof(tDataPath));

    strcpy(new_path->folder,target);
    curr_Path->next = new_path;
    new_path->prev = curr_Path;
    new_path->Head=head;
}

void Del_DataPath(tDataPath *curr_Path){
    curr_Path->prev->next=NULL;
}
```

OPER_cd

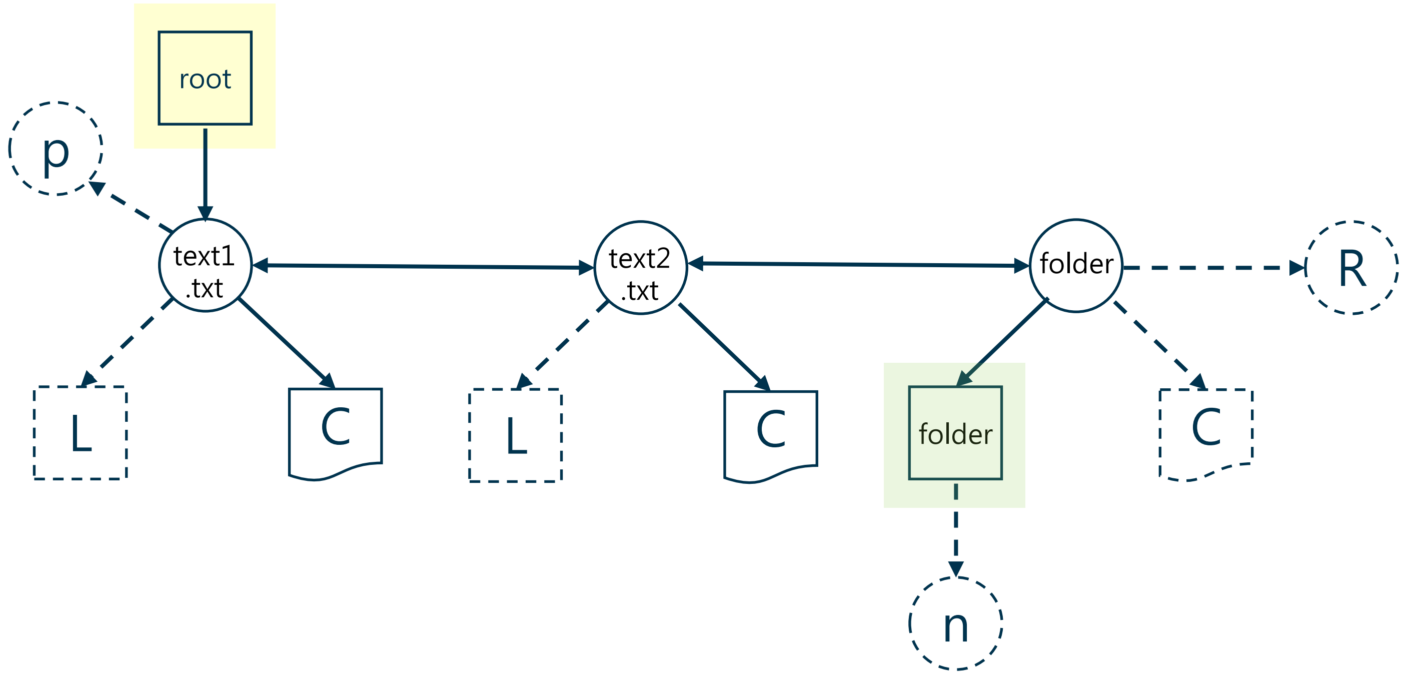
```
int OPER_cd(char target[],tDataPath *root,tDataPath *curr_Path);
FoR=OPER_cd(oper[1],root,curr_Path);
```

```
/ $ cd folder
/folder/ $ 
```

DataPath :

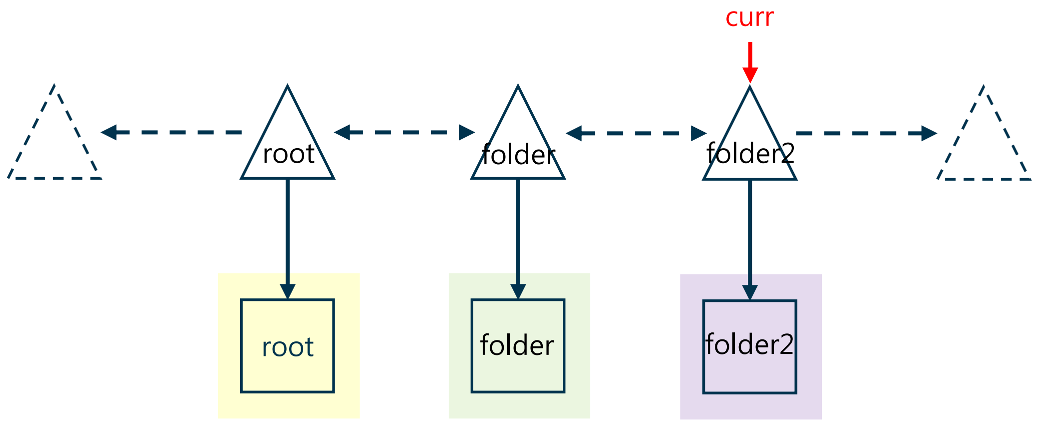


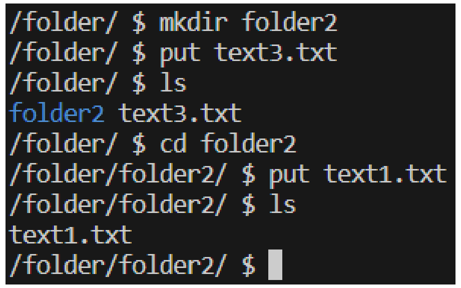
DataTree :



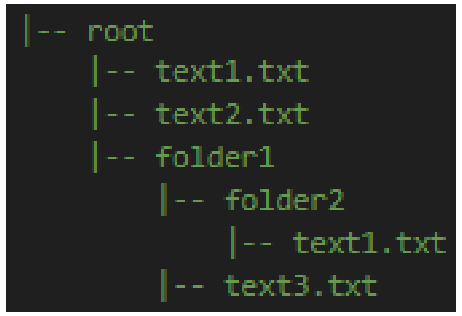
建立完成_DataPath

DataPath：





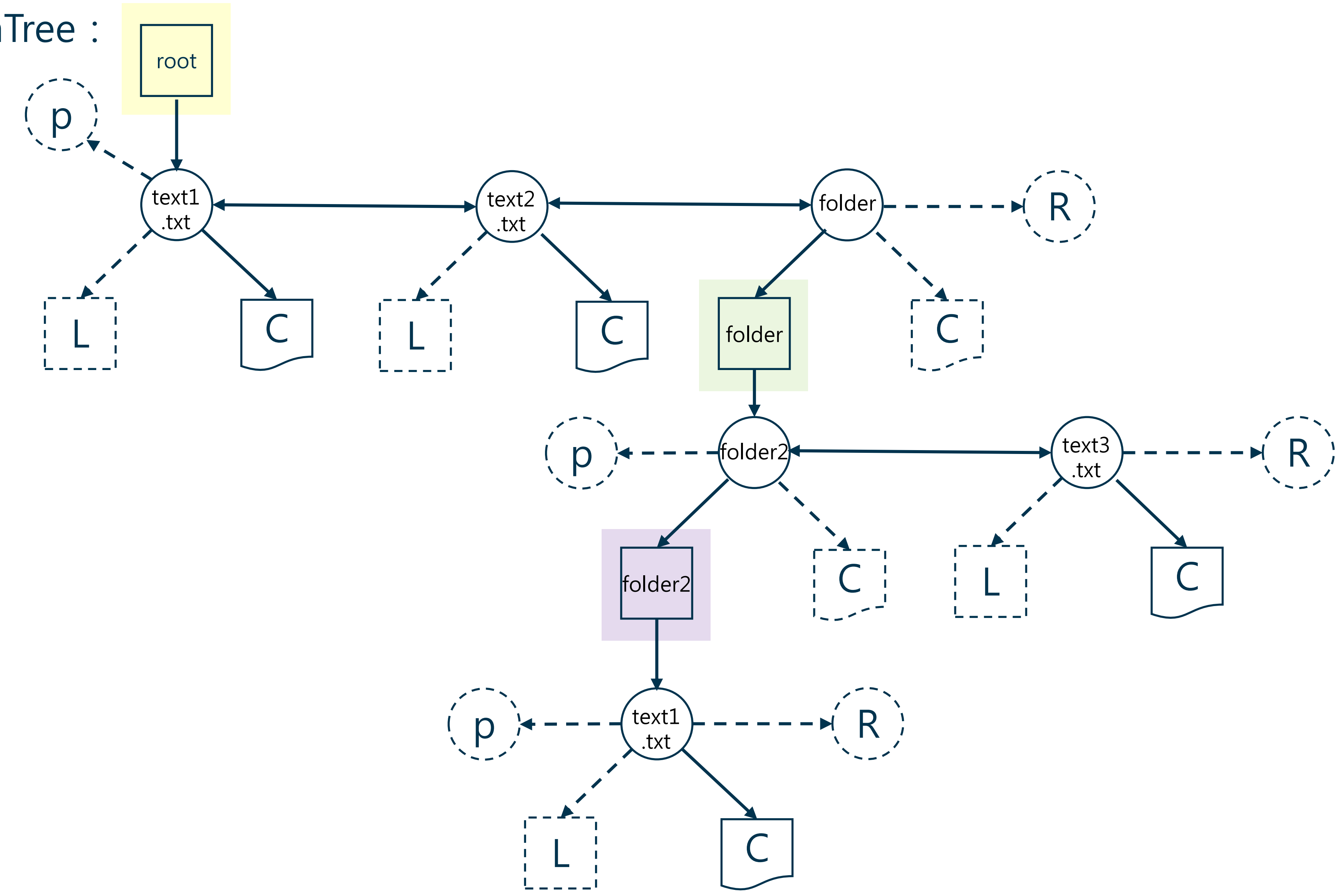
```
/folder/ $ mkdir folder2
/folder/ $ put text3.txt
/folder/ $ ls
folder2 text3.txt
/folder/ $ cd folder2
/folder/folder2/ $ put text1.txt
/folder/folder2/ $ ls
text1.txt
/folder/folder2/ $ 
```

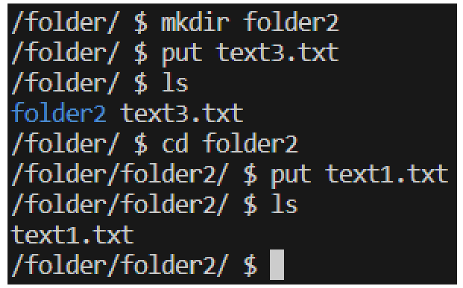


```
|-- root
    |-- text1.txt
    |-- text2.txt
    |-- folder1
        |-- folder2
            |-- text1.txt
        |-- text3.txt
```

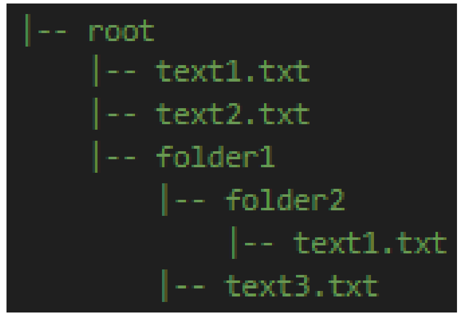
建立完成_DataTree

DataTree：





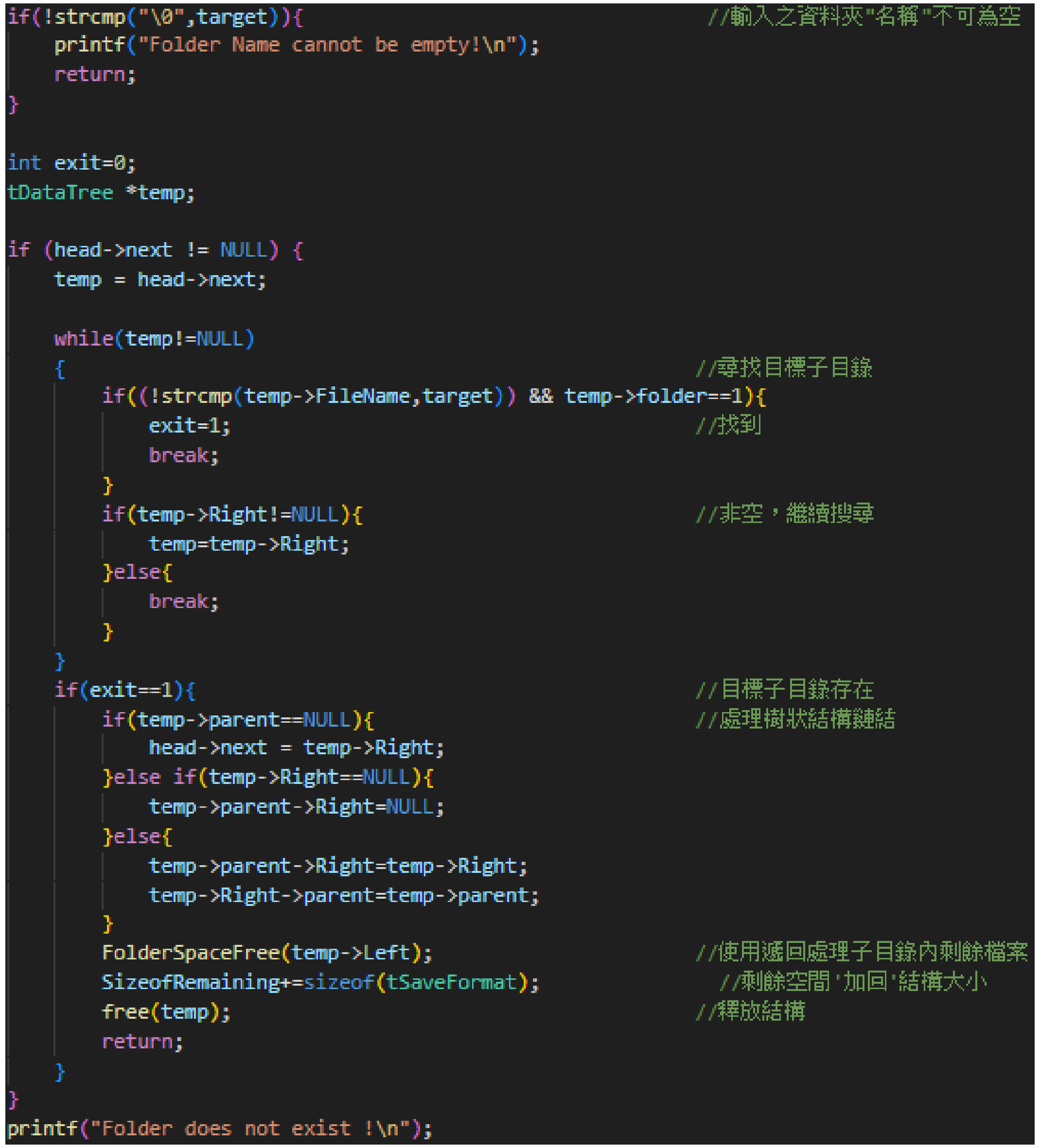
```
/folder/ $ mkdir folder2
/folder/ $ put text3.txt
/folder/ $ ls
folder2 text3.txt
/folder/ $ cd folder2
/folder/folder2/ $ put text1.txt
/folder/folder2/ $ ls
text1.txt
/folder/folder2/ $ 
```



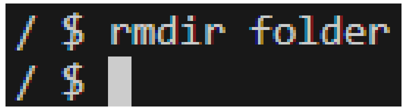
```
|-- root
    |-- text1.txt
    |-- text2.txt
    |-- folder1
        |-- folder2
            |-- text1.txt
        |-- text3.txt
```

OPER_rmdir

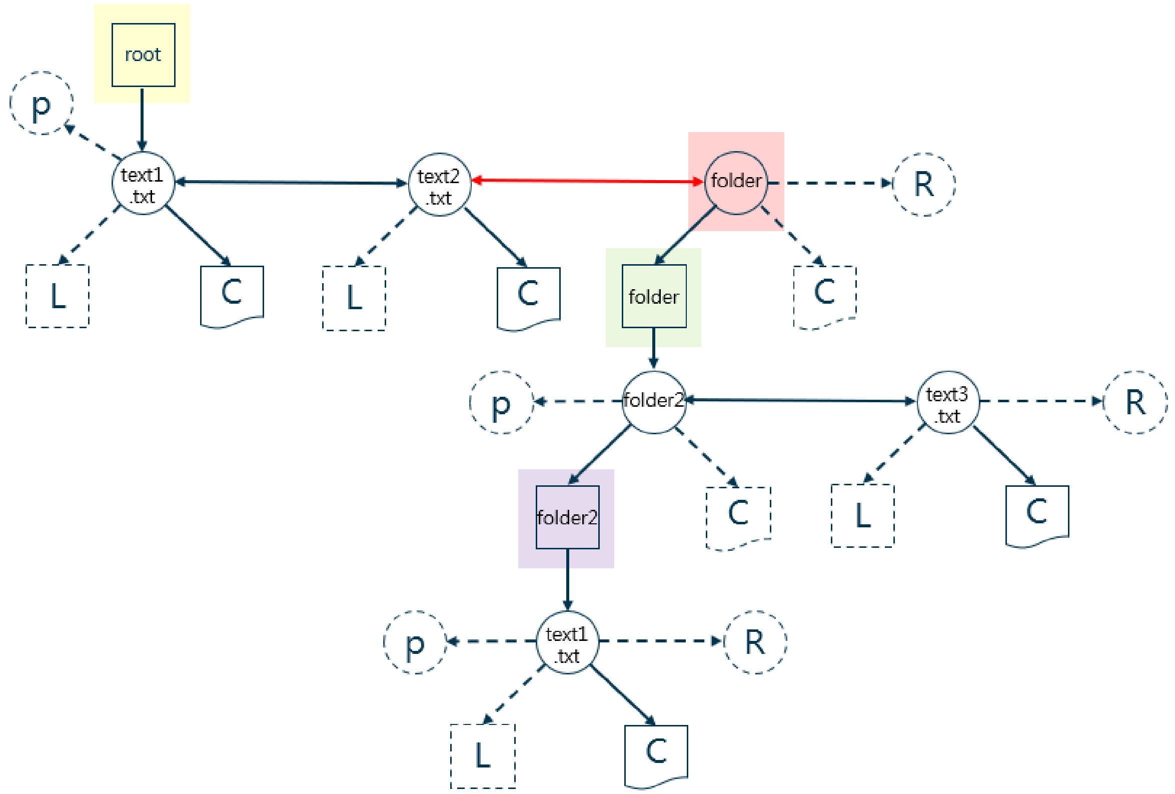
```
void OPER_rmdir(tDataHead *head,char target[]);
OPER_rmdir(curr_Path->Head,oper[1]);
```



```
if(!strcmp("\0",target)){                                           //輸入之資料夾"名稱"不可為空
    printf("Folder Name cannot be empty!\n");
    return;
}

int exit=0;
tDataTree *temp;

if (head->next != NULL) {
    temp = head->next;

    while(temp!=NULL)
    {                                                               //尋找目標子目錄
        if((!strcmp(temp->FileName,target)) && temp->folder==1){
            exit=1;                                                 //找到
            break;
        }
        if(temp->Right!=NULL){                                      //非空，繼續搜尋
            temp=temp->Right;
        }else{
            break;
        }
    }
    if(exit==1){                                                    //目標子目錄存在
        if(temp->parent==NULL){                                     //處理樹狀結構鏈結
            head->next = temp->Right;
        }else if(temp->Right==NULL){
            temp->parent->Right=NULL;
        }else{
            temp->parent->Right=temp->Right;
            temp->Right->parent=temp->parent;
        }
        FolderSpaceFree(temp->Left);                                //使用遞回處理子目錄內剩餘檔案
        SizeofRemaining+=sizeof(tSaveFormat);                         //剩餘空間'加回'結構大小
        free(temp);                                                 //釋放結構
        return;
    }
}
printf("Folder does not exist !\n");
```

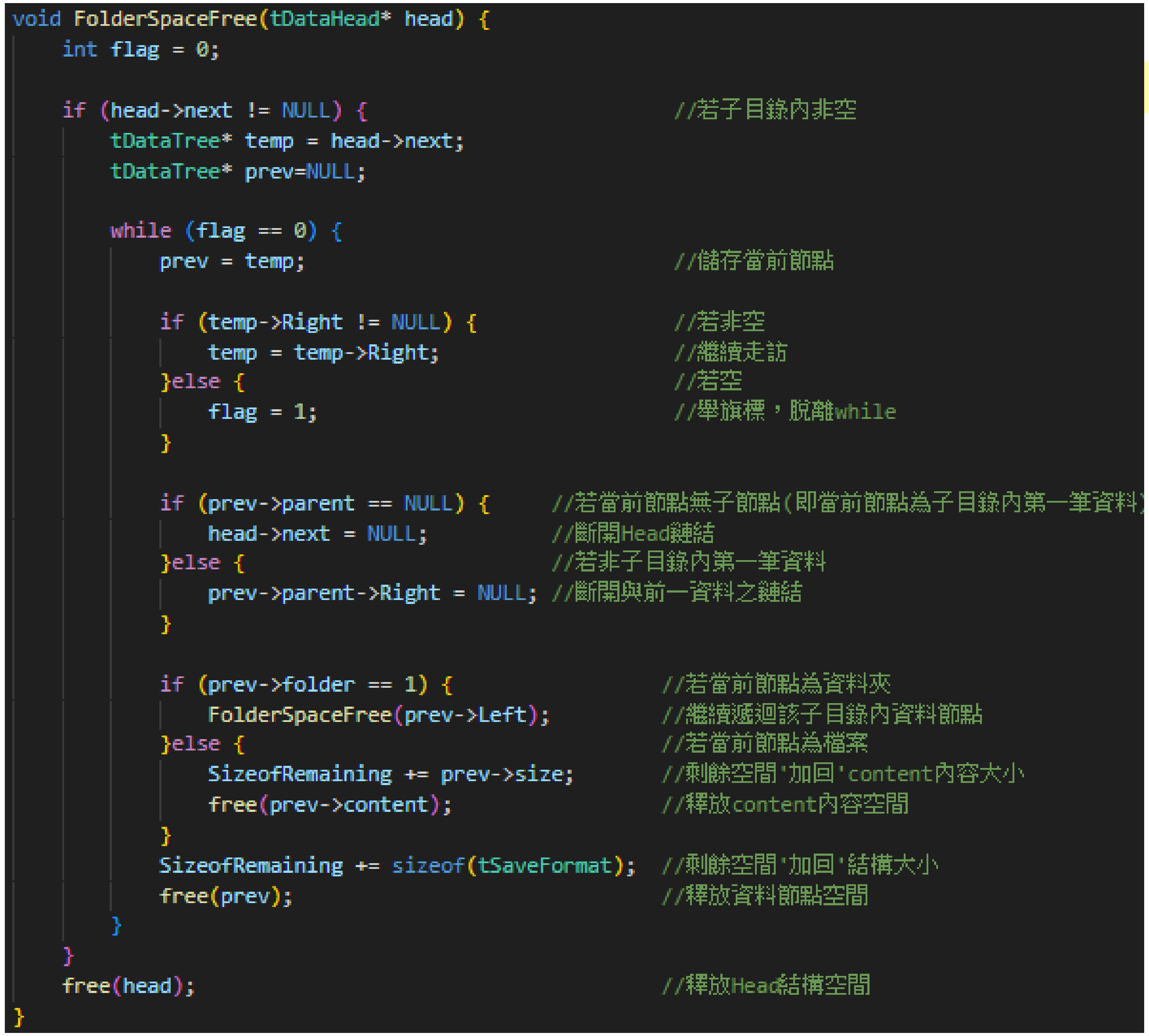


```
/ $ rmdir folder
/ $ 
```

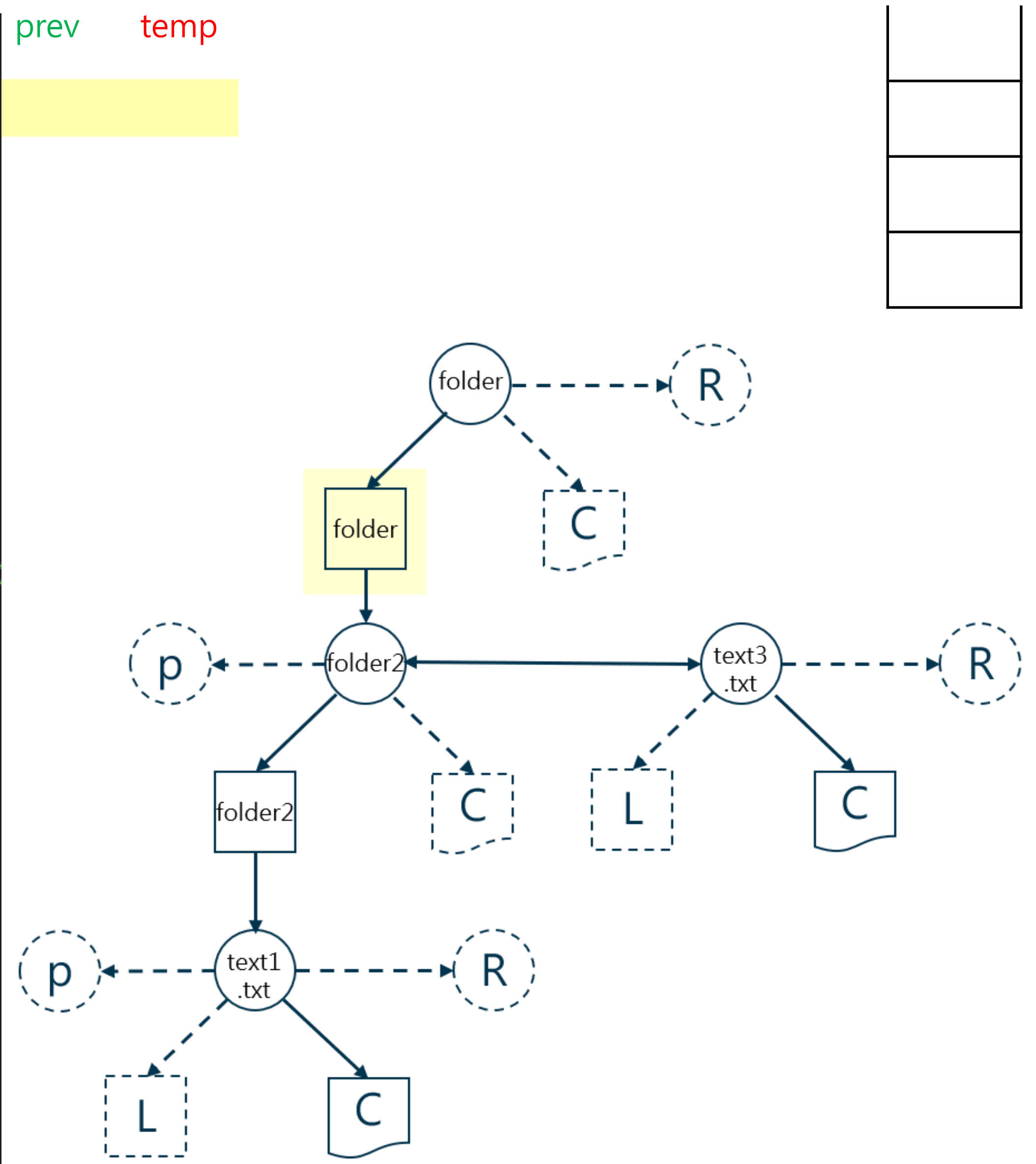


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

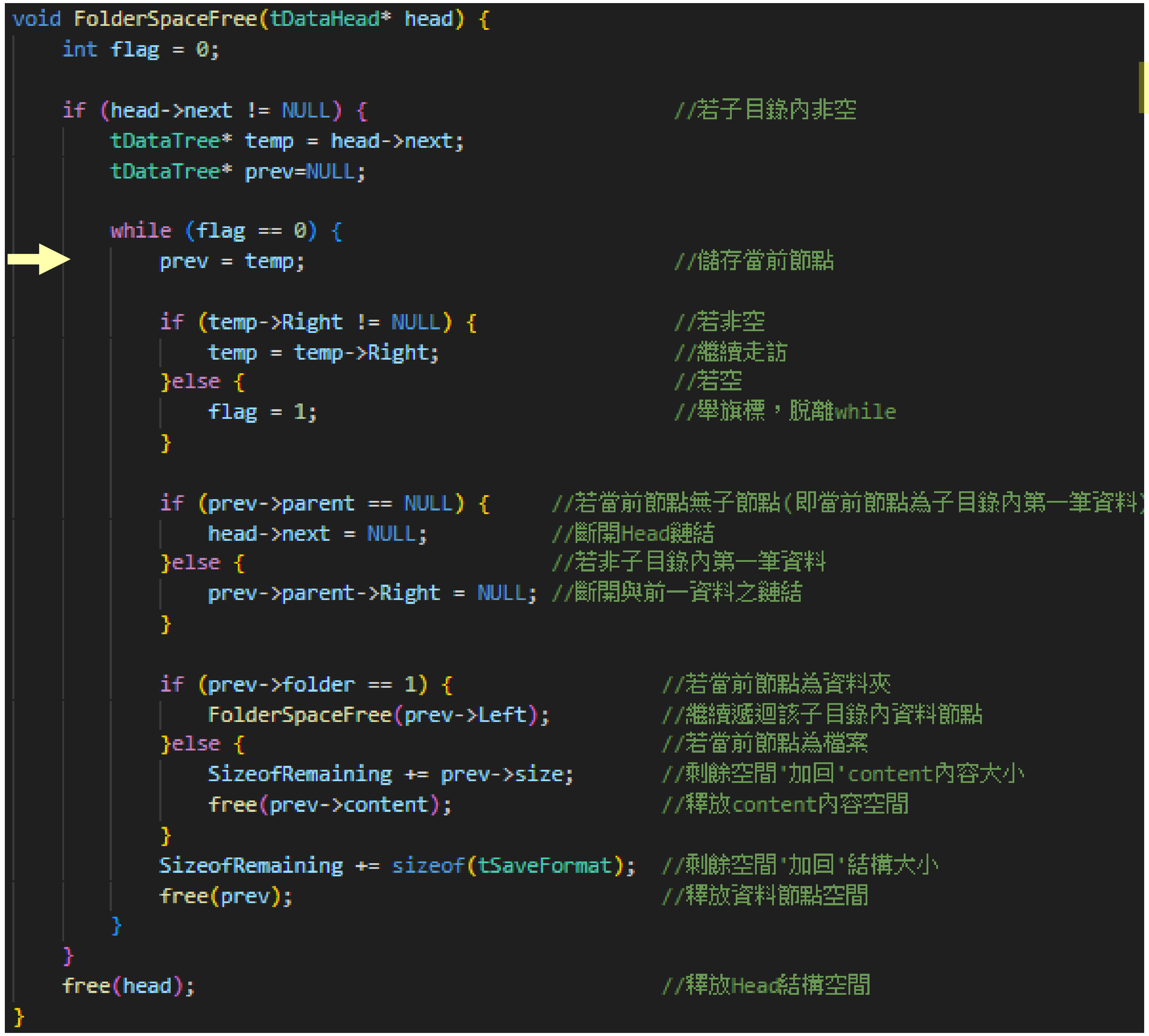


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                              //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                                   //儲存當前節點

            if (temp->Right != NULL) {                     //若非空
                temp = temp->Right;                        //繼續走訪
            }else {                                        //若空
                flag = 1;                                  //舉旗標，脫離while
            }

            if (prev->parent == NULL) {           //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;                //斷開Head鏈結
            }else {                               //若非子目錄內第一筆資料
                prev->parent->Right = NULL;       //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                      //若當前節點為資料夾
                FolderSpaceFree(prev->Left);              //繼續遞迴該子目錄內資料節點
            }else {                                       //若當前節點為檔案
                SizeofRemaining += prev->size;            //剩餘空間'加回'content內容大小
                free(prev->content);                      //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);       //剩餘空間'加回'結構大小
            free(prev);                                   //釋放資料節點空間
        }
    }
    free(head);                                           //釋放Head結構空間
}
```

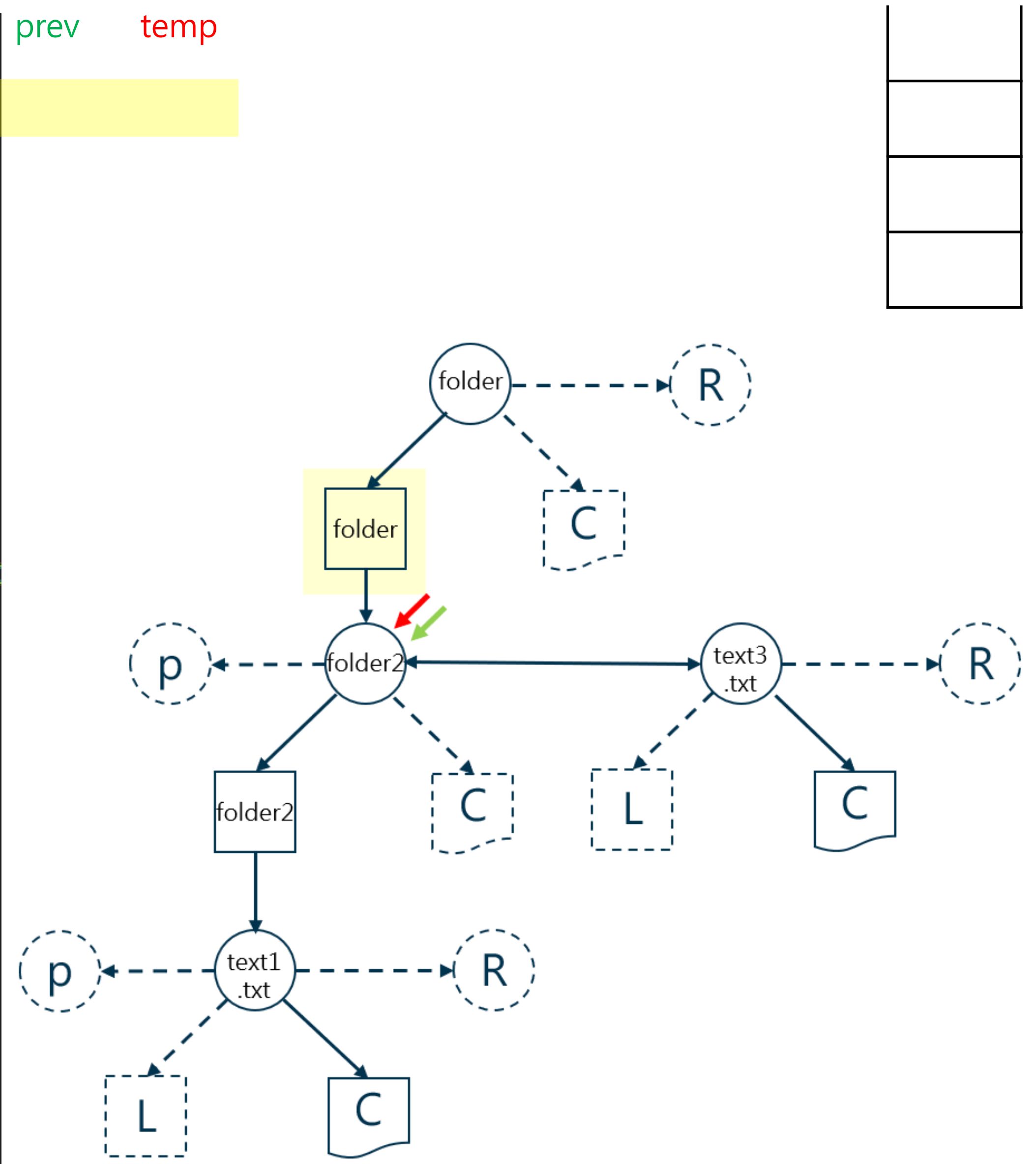


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

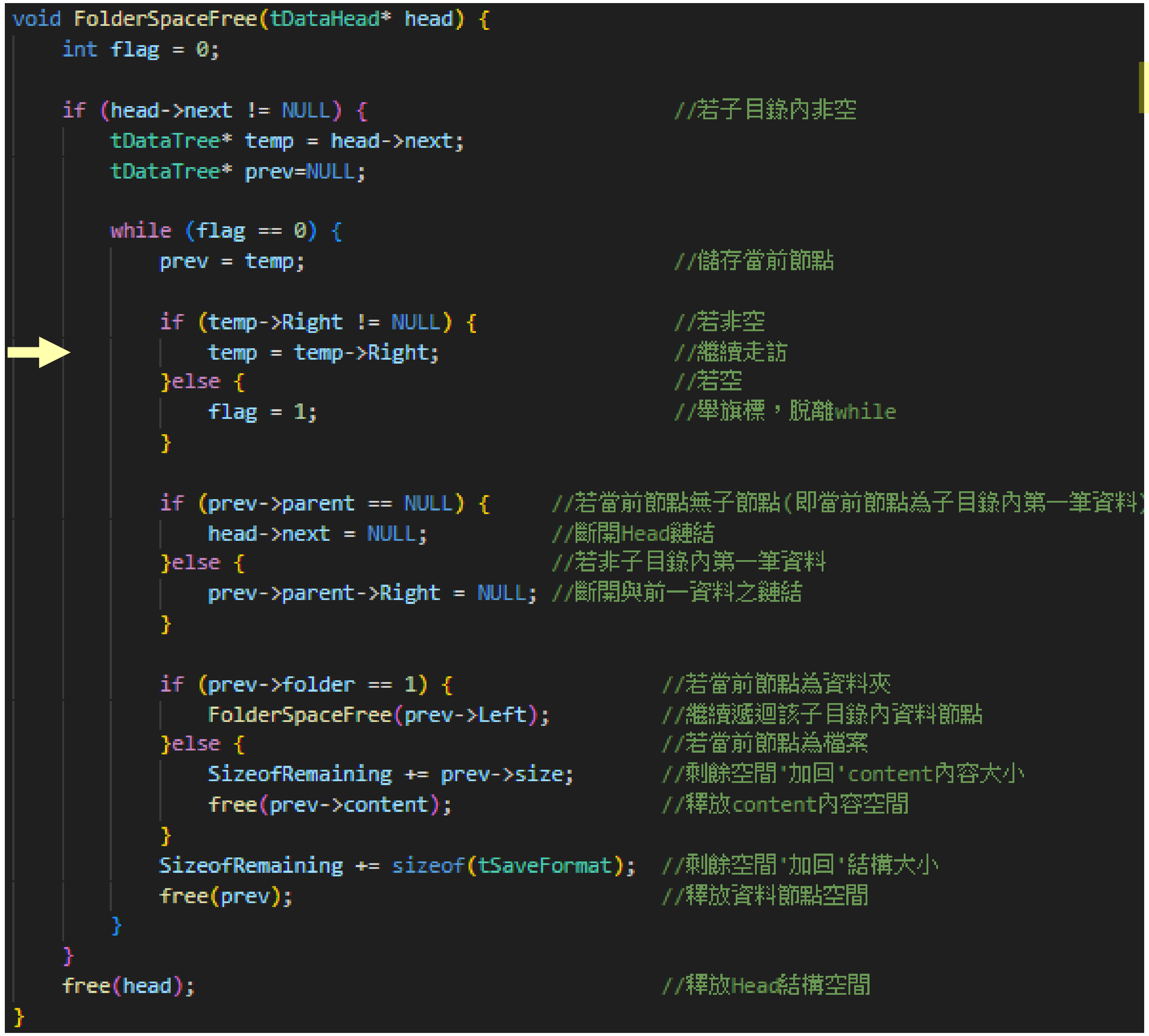


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                         //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                              //儲存當前節點

            if (temp->Right != NULL) {                //若非空
                temp = temp->Right;                   //繼續走訪
            }else {                                   //若空
                flag = 1;                             //舉旗標，脫離while
            }

            if (prev->parent == NULL) {      //若當前節點無子節點(即當前節點為子目錄內第一筆資料
                head->next = NULL;           //斷開Head鏈結
            }else {                          //若非子目錄內第一筆資料
                prev->parent->Right = NULL;  //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                 //若當前節點為資料夾
                FolderSpaceFree(prev->Left);         //繼續遞迴該子目錄內資料節點
            }else {                                  //若當前節點為檔案
                SizeofRemaining += prev->size;       //剩餘空間'加回'content內容大小
                free(prev->content);                 //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);  //剩餘空間'加回'結構大小
            free(prev);                              //釋放資料節點空間
        }
    }
    free(head);                                      //釋放Head結構空間
}
```

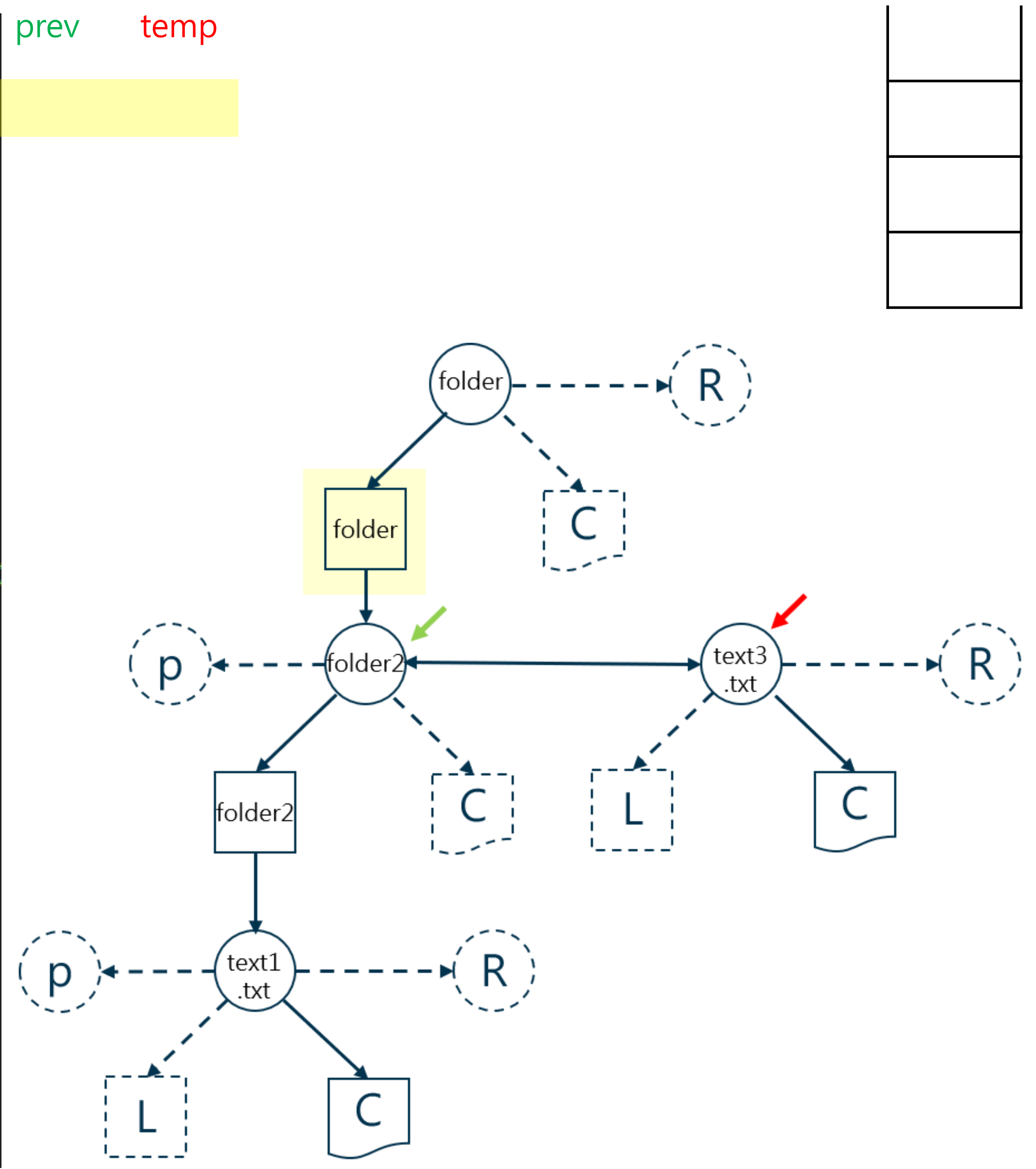


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

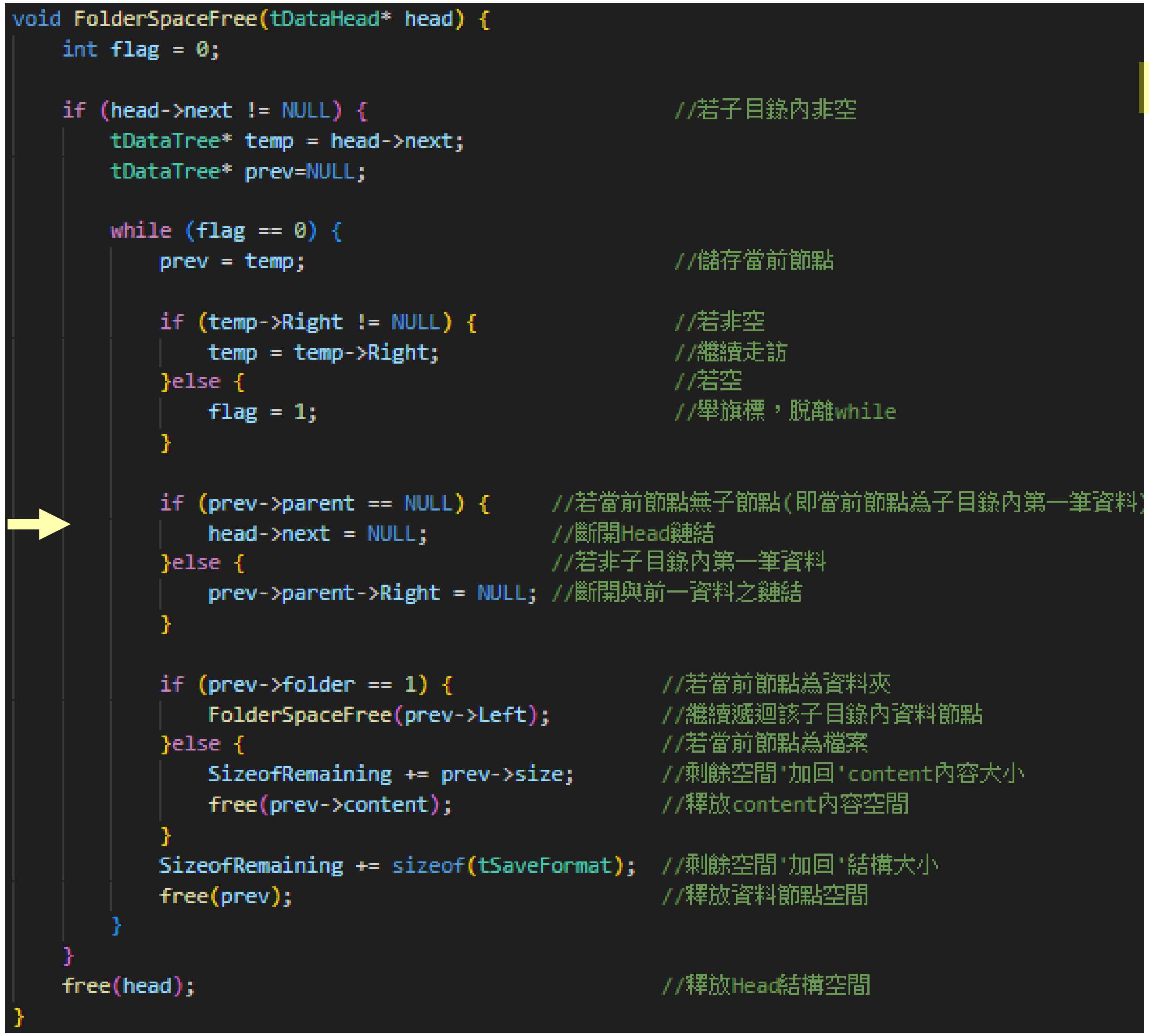


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                        //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                             //儲存當前節點

            if (temp->Right != NULL) {               //若非空
                temp = temp->Right;                  //繼續走訪
            }else {                                  //若空
                flag = 1;                            //舉旗標，脫離while
            }

            if (prev->parent == NULL) {      //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;           //斷開Head鏈結
            }else {                          //若非子目錄內第一筆資料
                prev->parent->Right = NULL; //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                //若當前節點為資料夾
                FolderSpaceFree(prev->Left);        //繼續遞迴該子目錄內資料節點
            }else {                                 //若當前節點為檔案
                SizeofRemaining += prev->size;      //剩餘空間'加回'content內容大小
                free(prev->content);                //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat); //剩餘空間'加回'結構大小
            free(prev);                             //釋放資料節點空間
        }
    }
    free(head);                                     //釋放Head結構空間
}
```



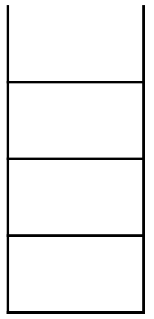
FolderSpaceFree

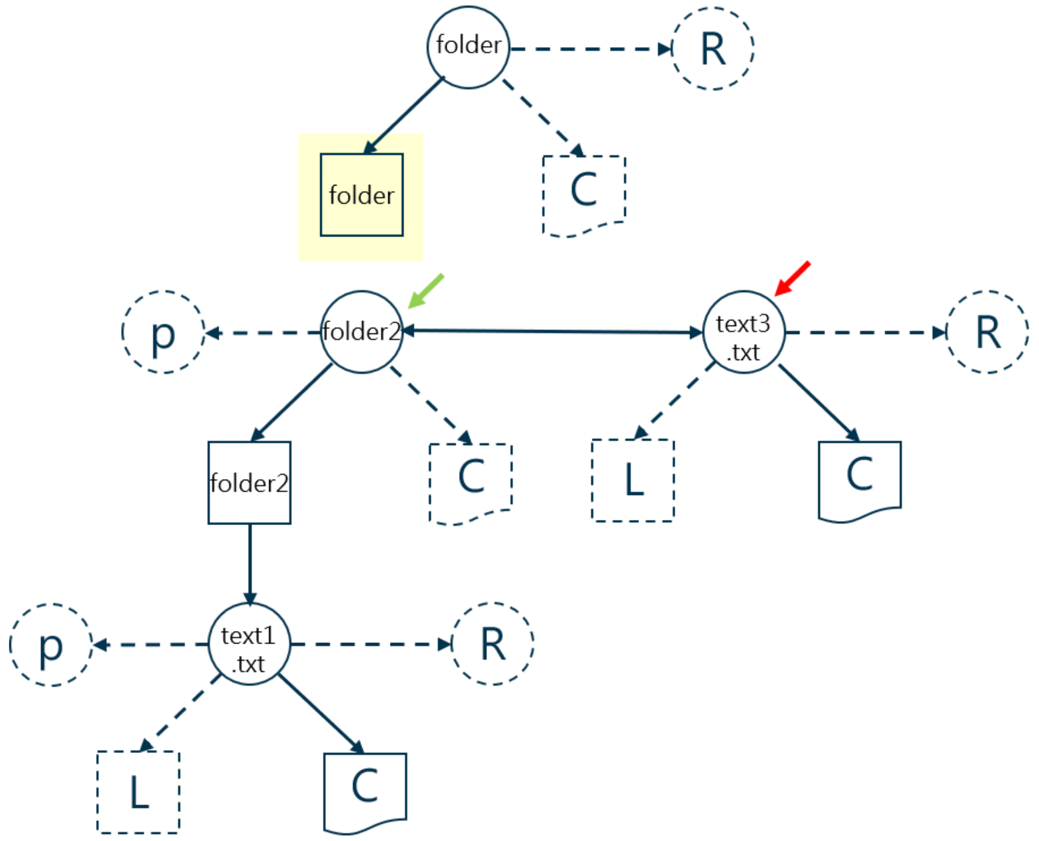
```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```



```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                         //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                              //儲存當前節點

            if (temp->Right != NULL) {                //若非空
                temp = temp->Right;                   //繼續走訪
            }else {                                   //若空
                flag = 1;                             //舉旗標，脫離while
            }

            if (prev->parent == NULL) {      //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;           //斷開Head鏈結
            }else {                          //若非子目錄內第一筆資料
                prev->parent->Right = NULL;  //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                 //若當前節點為資料夾
                FolderSpaceFree(prev->Left);         //繼續遞迴該子目錄內資料節點
            }else {                                  //若當前節點為檔案
                SizeofRemaining += prev->size;       //剩餘空間'加回'content內容大小
                free(prev->content);                 //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);  //剩餘空間'加回'結構大小
            free(prev);                              //釋放資料節點空間
        }
    }
    free(head);                                      //釋放Head結構空間
}
```

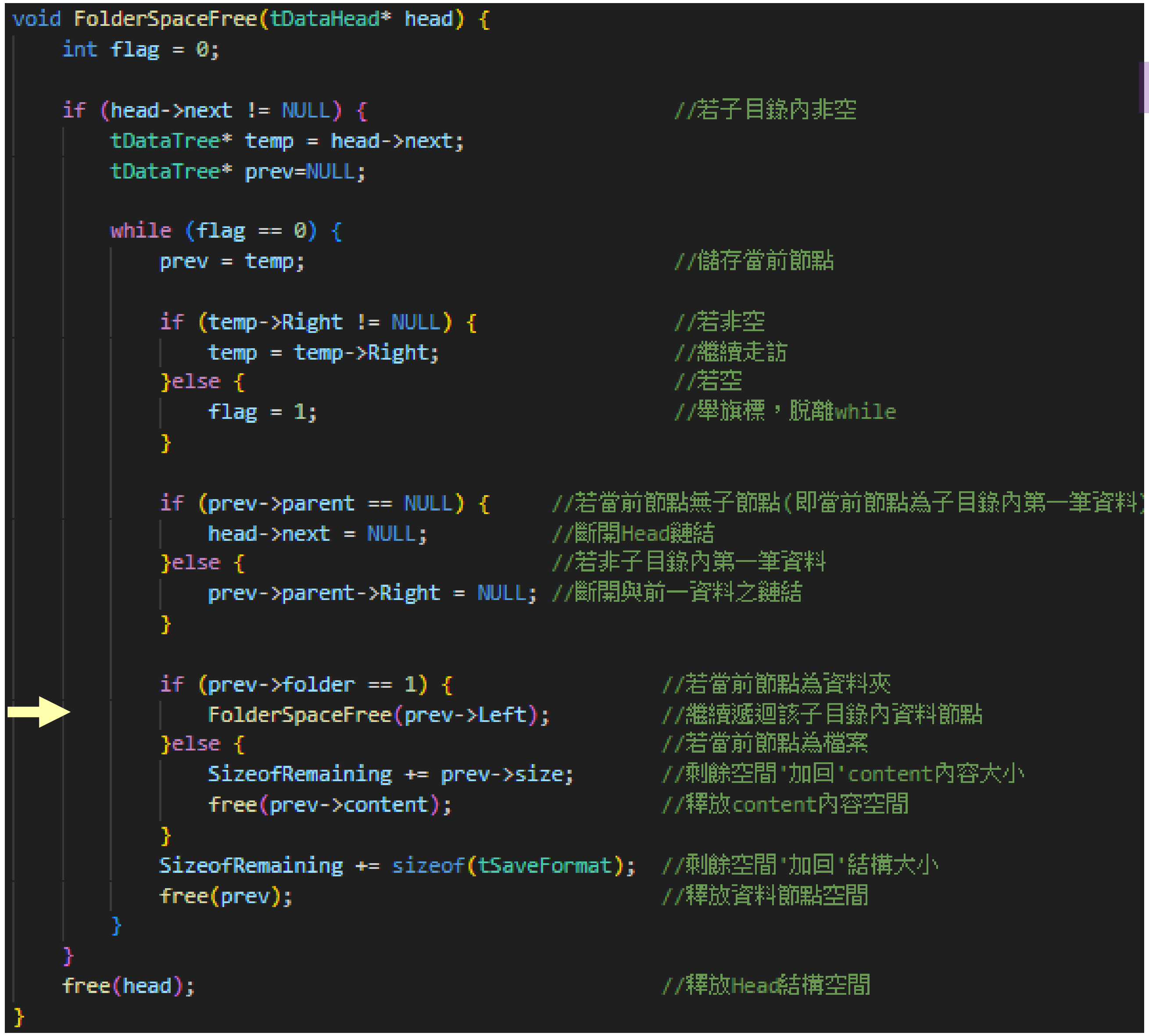
prev temp



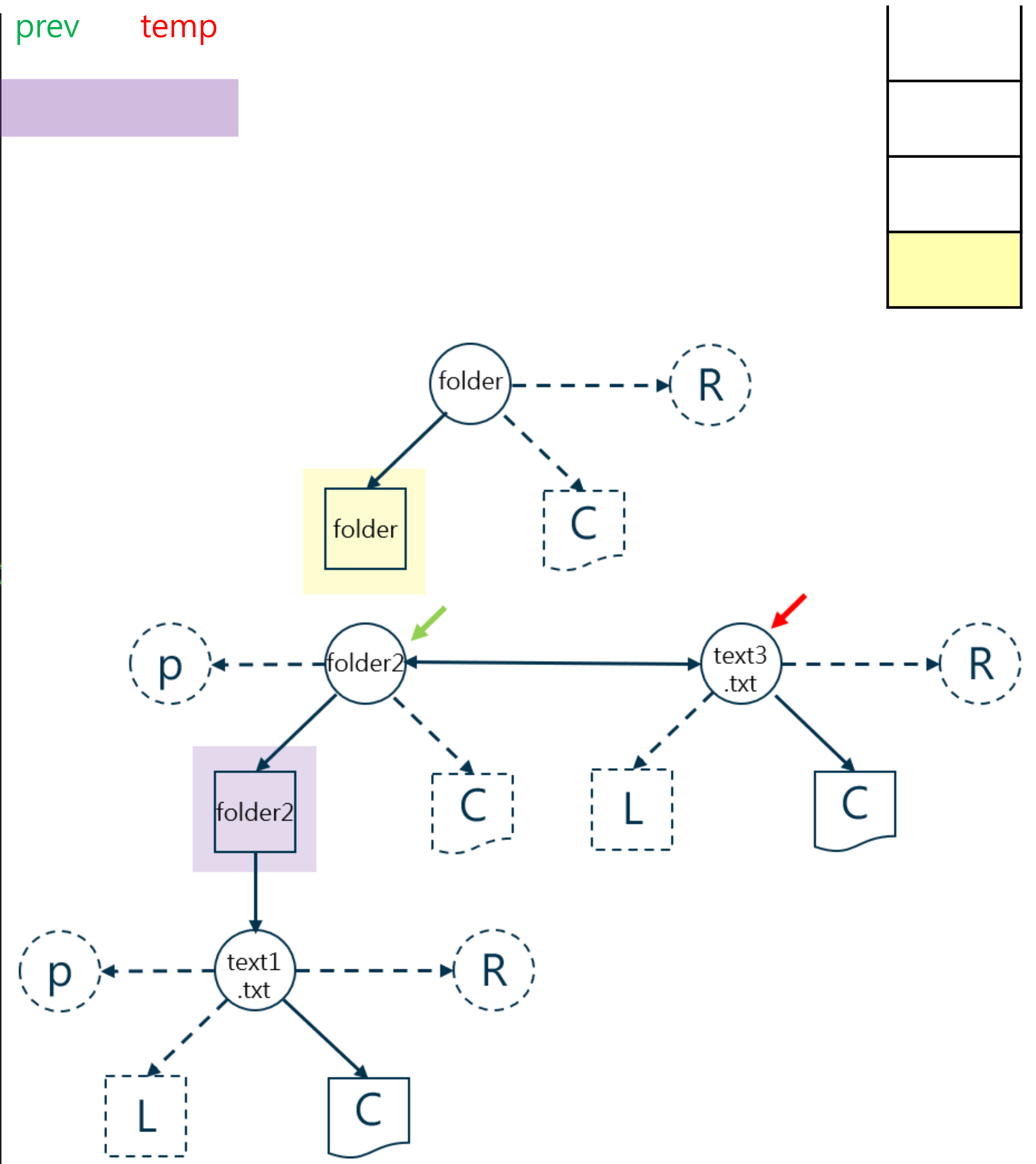


FolderSpaceFree

```c
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

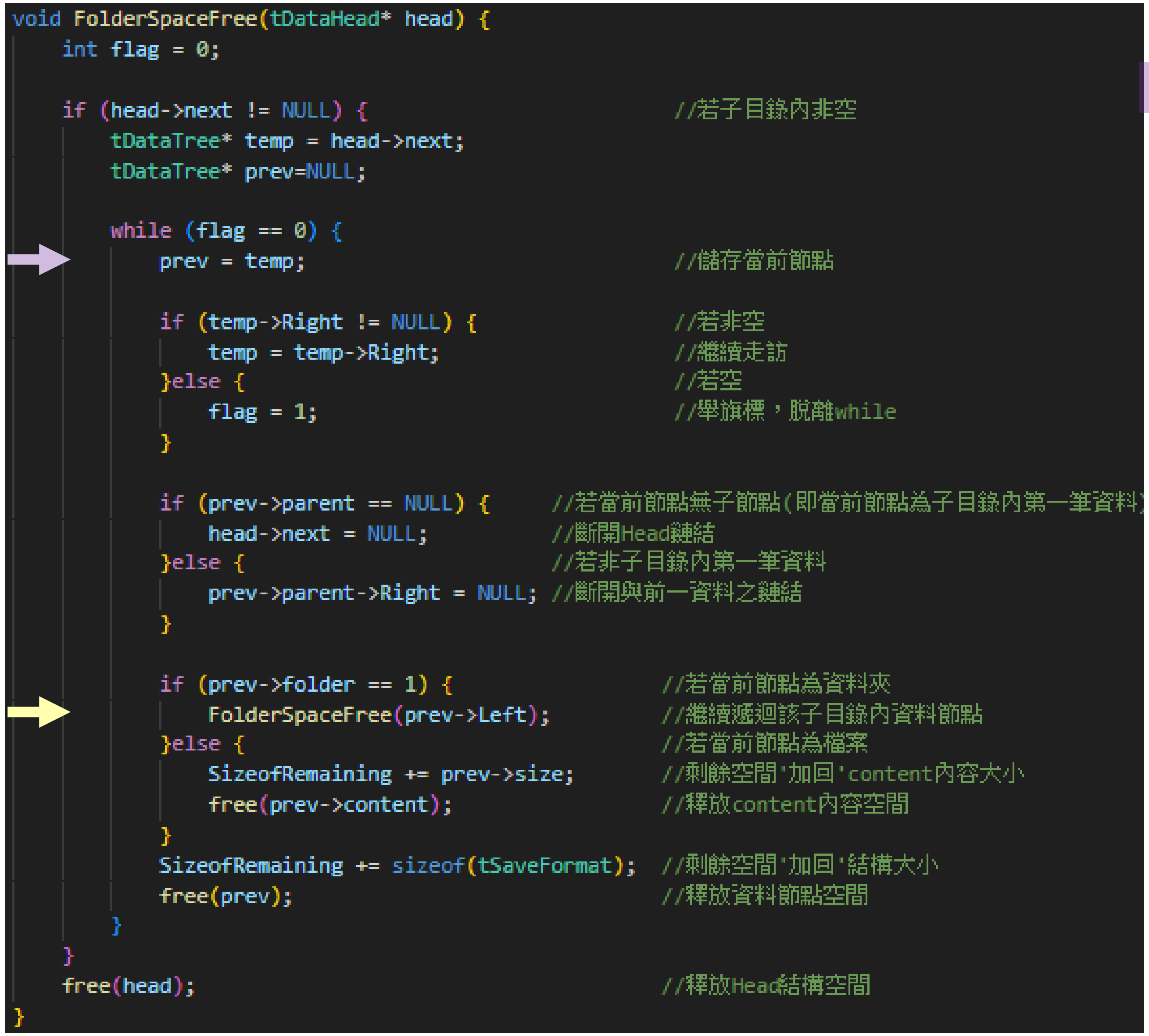


```c
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                        //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                             //儲存當前節點

            if (temp->Right != NULL) {               //若非空
                temp = temp->Right;                  //繼續走訪
            }else {                                  //若空
                flag = 1;                            //舉旗標，脫離while
            }

            if (prev->parent == NULL) {      //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;           //斷開Head鏈結
            }else {                          //若非子目錄內第一筆資料
                prev->parent->Right = NULL; //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                //若當前節點為資料夾
                FolderSpaceFree(prev->Left);        //繼續遞迴該子目錄內資料節點
            }else {                                 //若當前節點為檔案
                SizeofRemaining += prev->size;      //剩餘空間'加回'content內容大小
                free(prev->content);                //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat); //剩餘空間'加回'結構大小
            free(prev);                             //釋放資料節點空間
        }
    }
    free(head);                                     //釋放Head結構空間
}
```

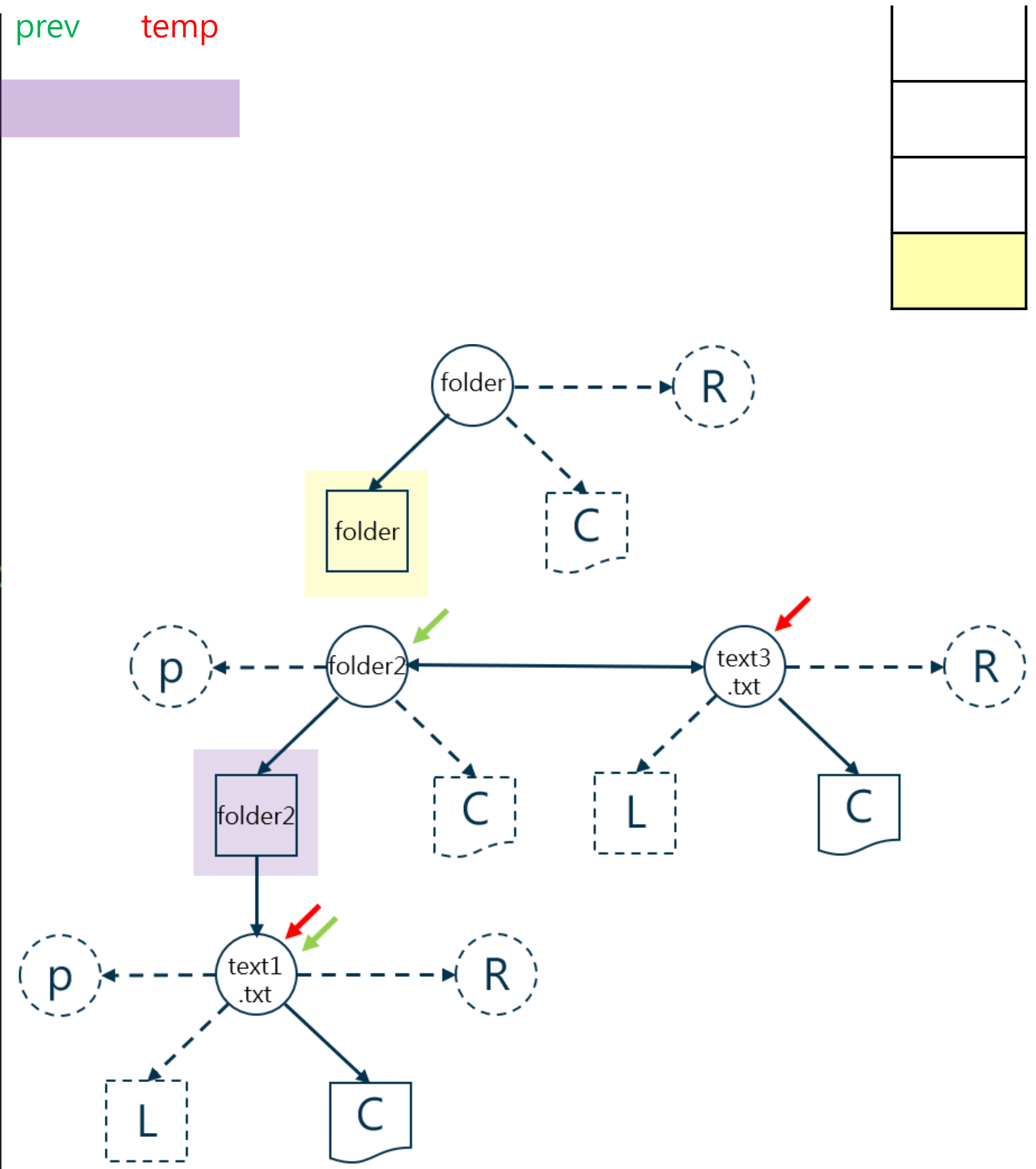


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

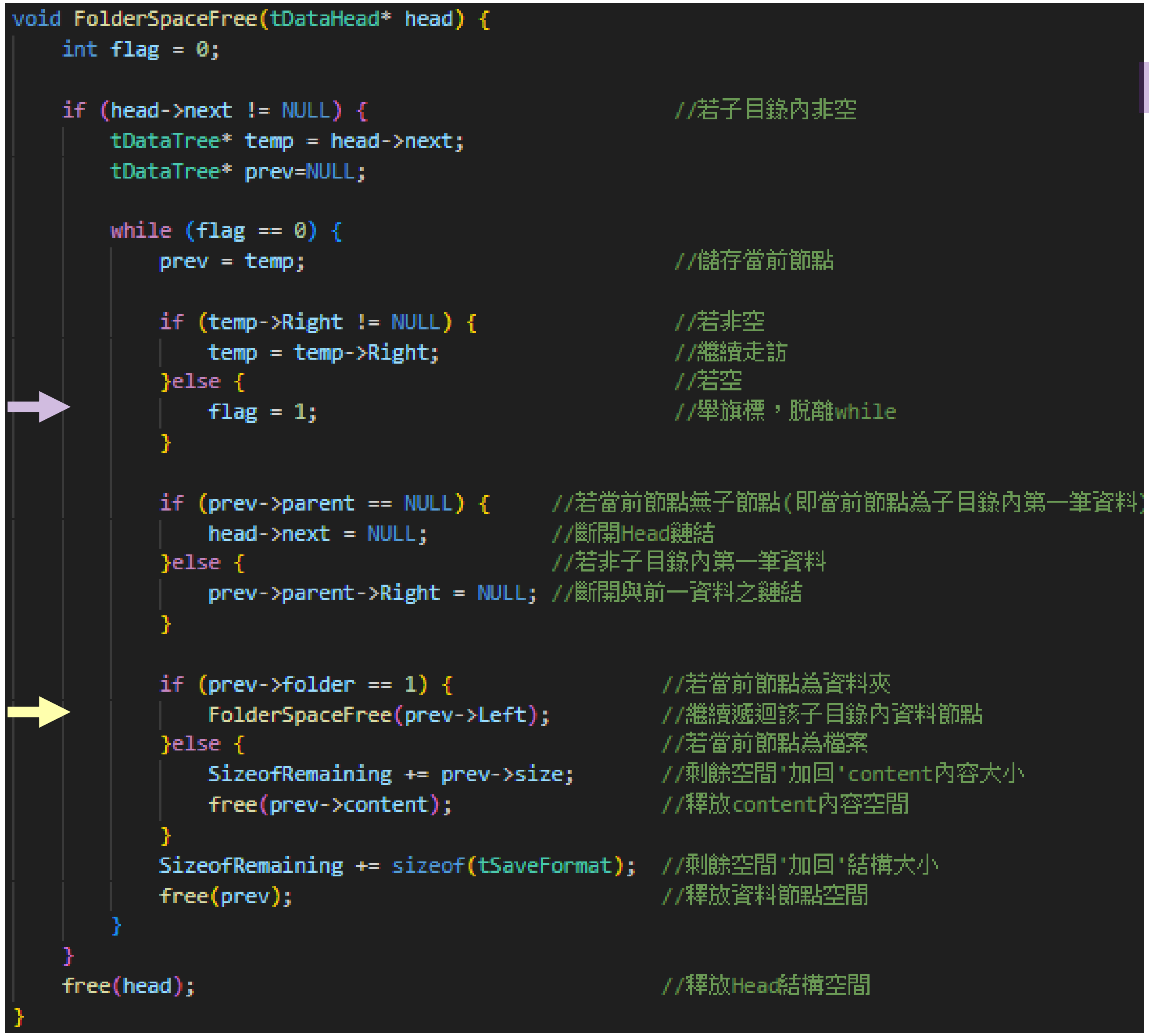


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                  //若當前節點為資料夾
                FolderSpaceFree(prev->Left);          //繼續遞迴該子目錄內資料節點
            }else {                                   //若當前節點為檔案
                SizeofRemaining += prev->size;        //剩餘空間'加回'content內容大小
                free(prev->content);                  //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);   //剩餘空間'加回'結構大小
            free(prev);                               //釋放資料節點空間
        }
    }
    free(head);                                       //釋放Head結構空間
}
```

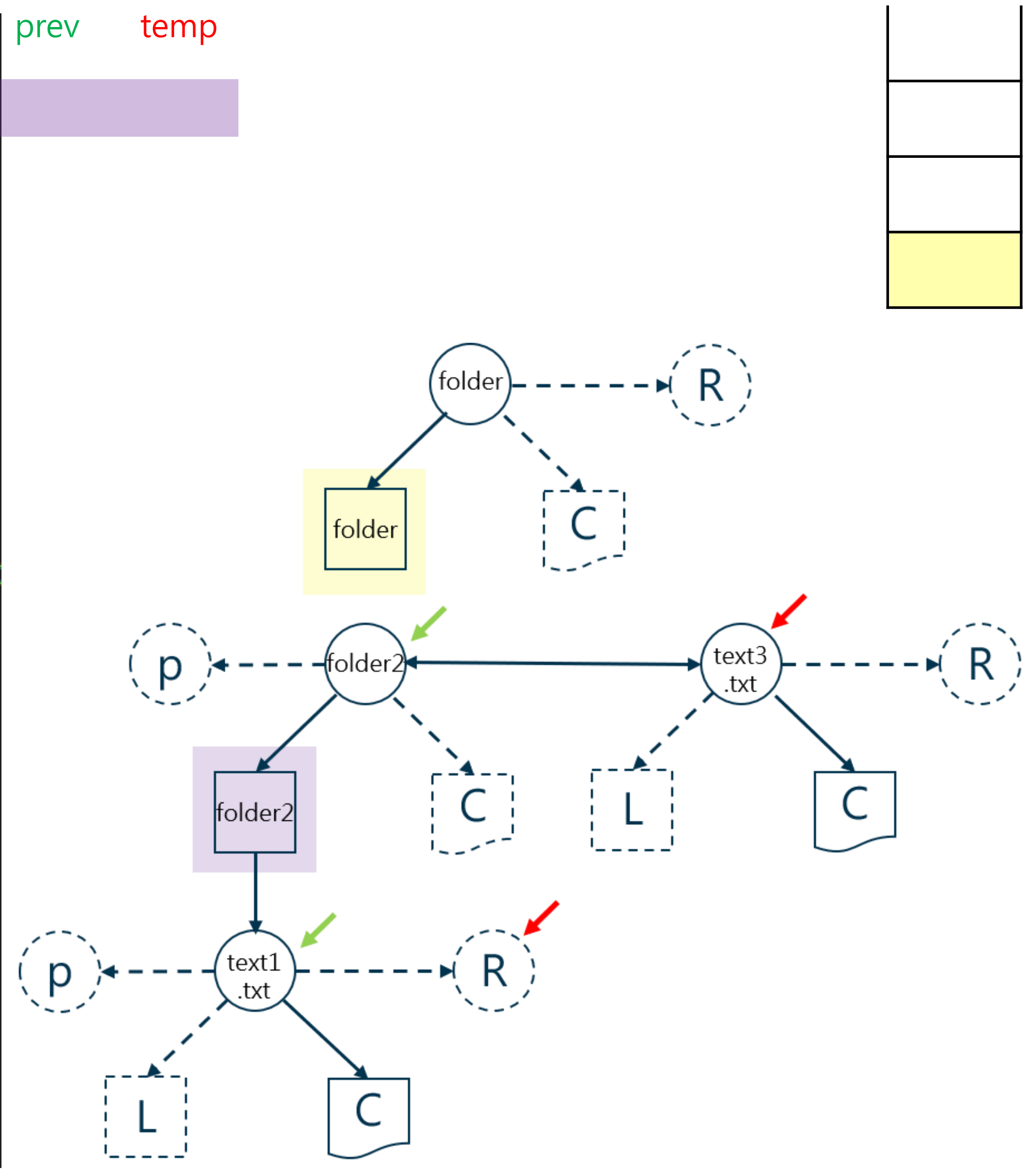


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

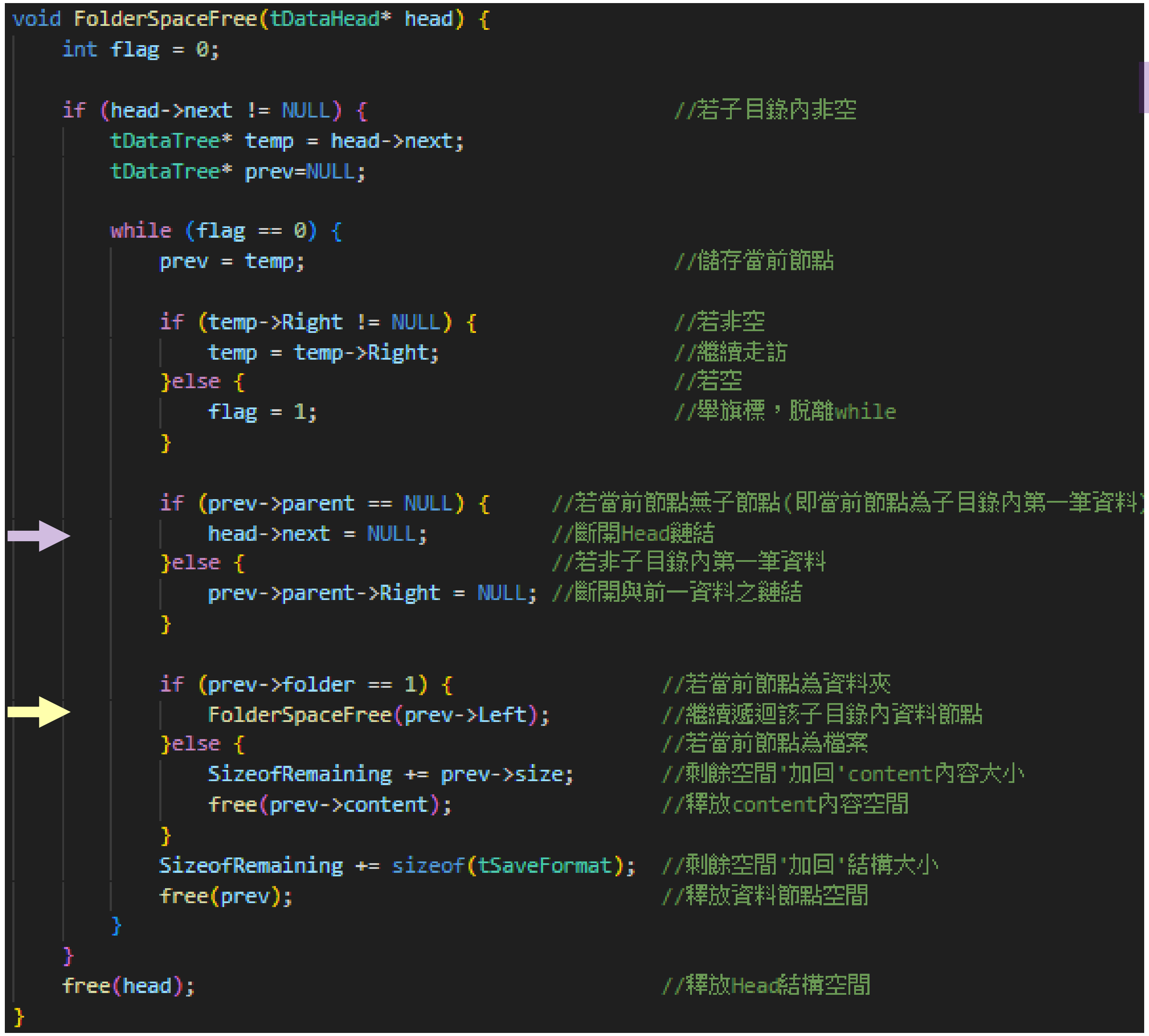


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {      //若當前節點無子節點(即當前節點為子目錄內第一筆資料
                head->next = NULL;           //斷開Head鏈結
            }else {                          //若非子目錄內第一筆資料
                prev->parent->Right = NULL;  //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                  //若當前節點為資料夾
                FolderSpaceFree(prev->Left);          //繼續遞迴該子目錄內資料節點
            }else {                                   //若當前節點為檔案
                SizeofRemaining += prev->size;        //剩餘空間'加回'content內容大小
                free(prev->content);                  //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);  //剩餘空間'加回'結構大小
            free(prev);                               //釋放資料節點空間
        }
    }
    free(head);                                       //釋放Head結構空間
}
```

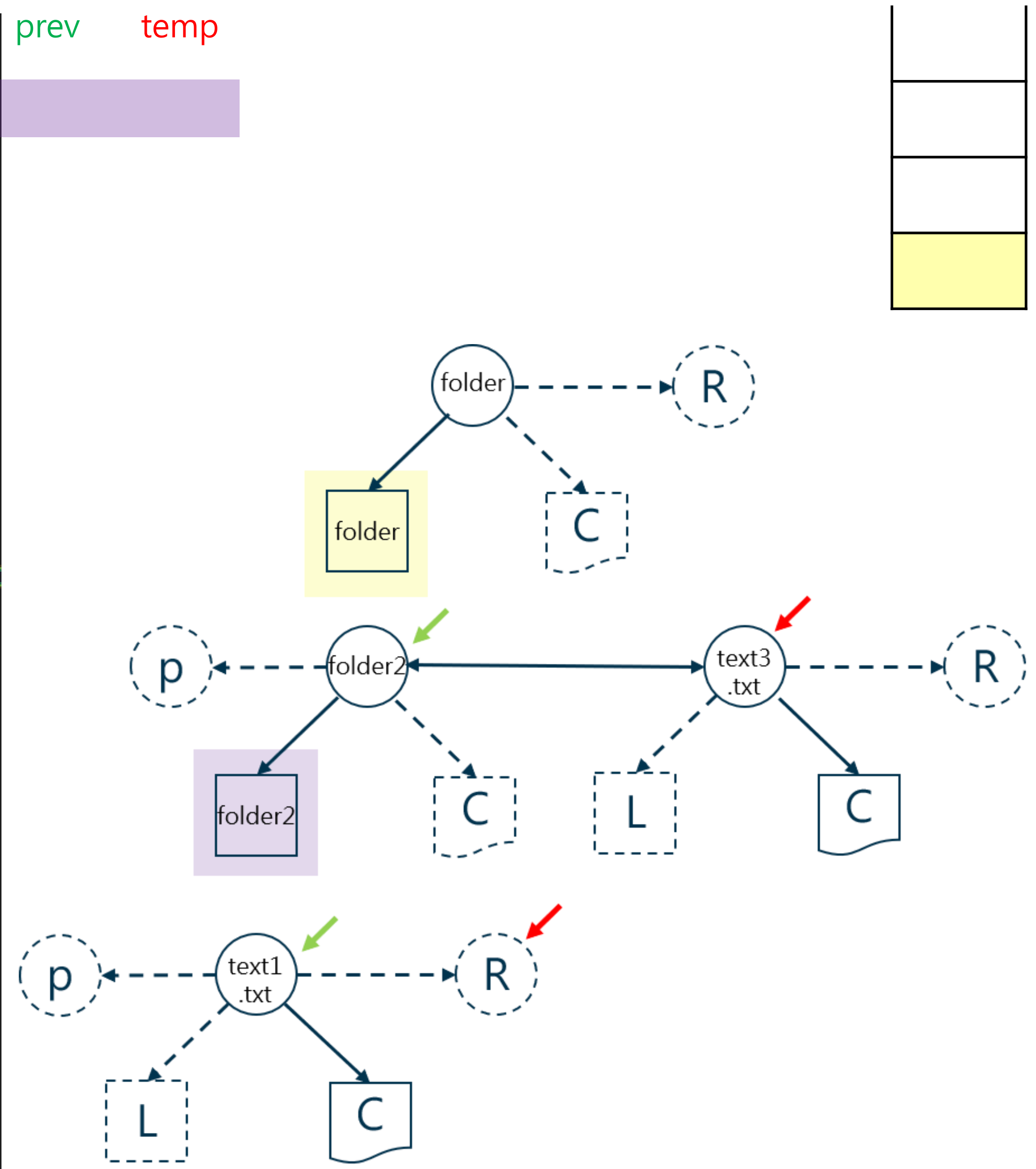


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

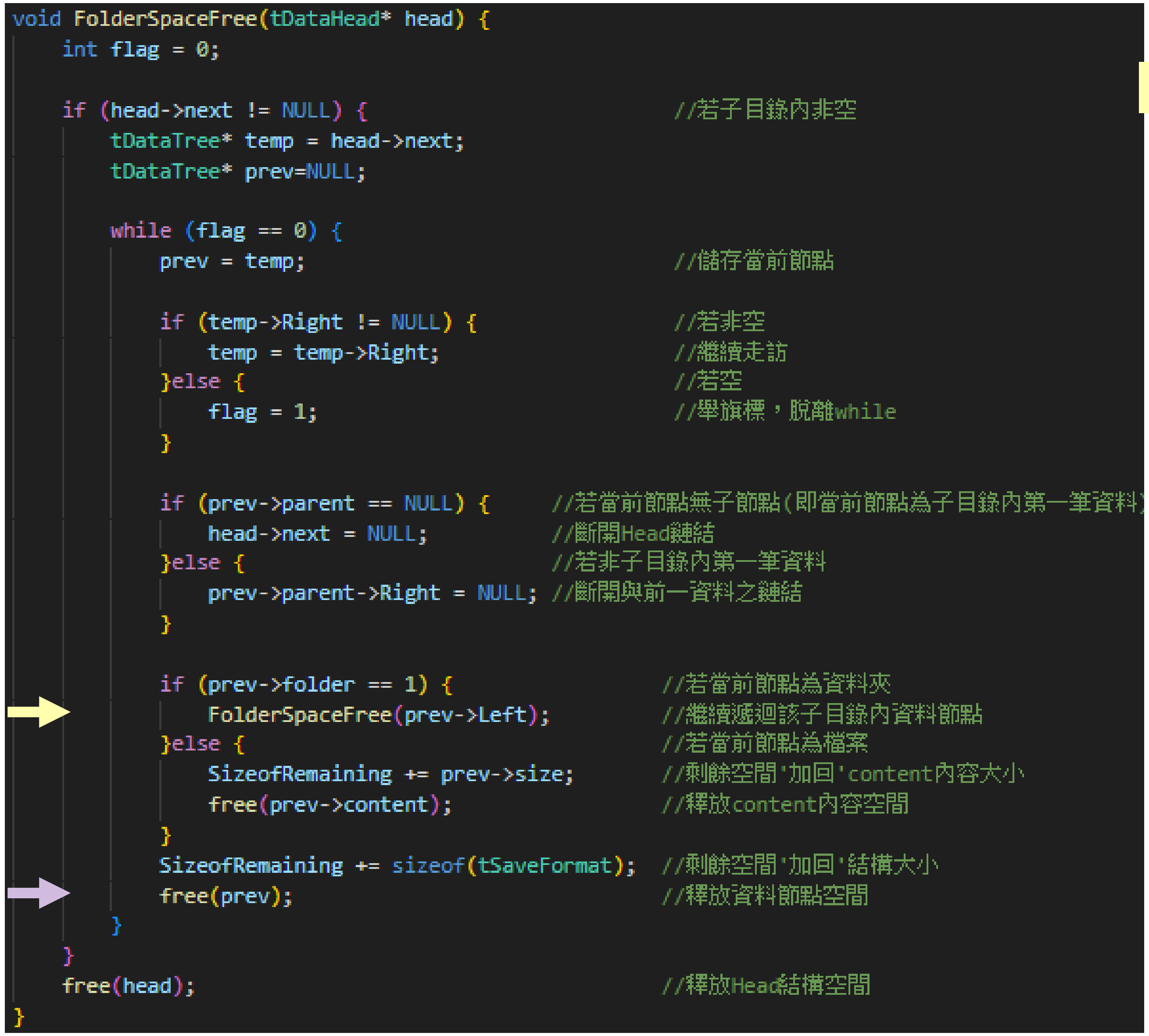


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                 //若當前節點為資料夾
                FolderSpaceFree(prev->Left);         //繼續遞迴該子目錄內資料節點
            }else {                                  //若當前節點為檔案
                SizeofRemaining += prev->size;       //剩餘空間'加回'content內容大小
                free(prev->content);                 //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);  //剩餘空間'加回'結構大小
            free(prev);                              //釋放資料節點空間
        }
    }
    free(head);                                      //釋放Head結構空間
}
```

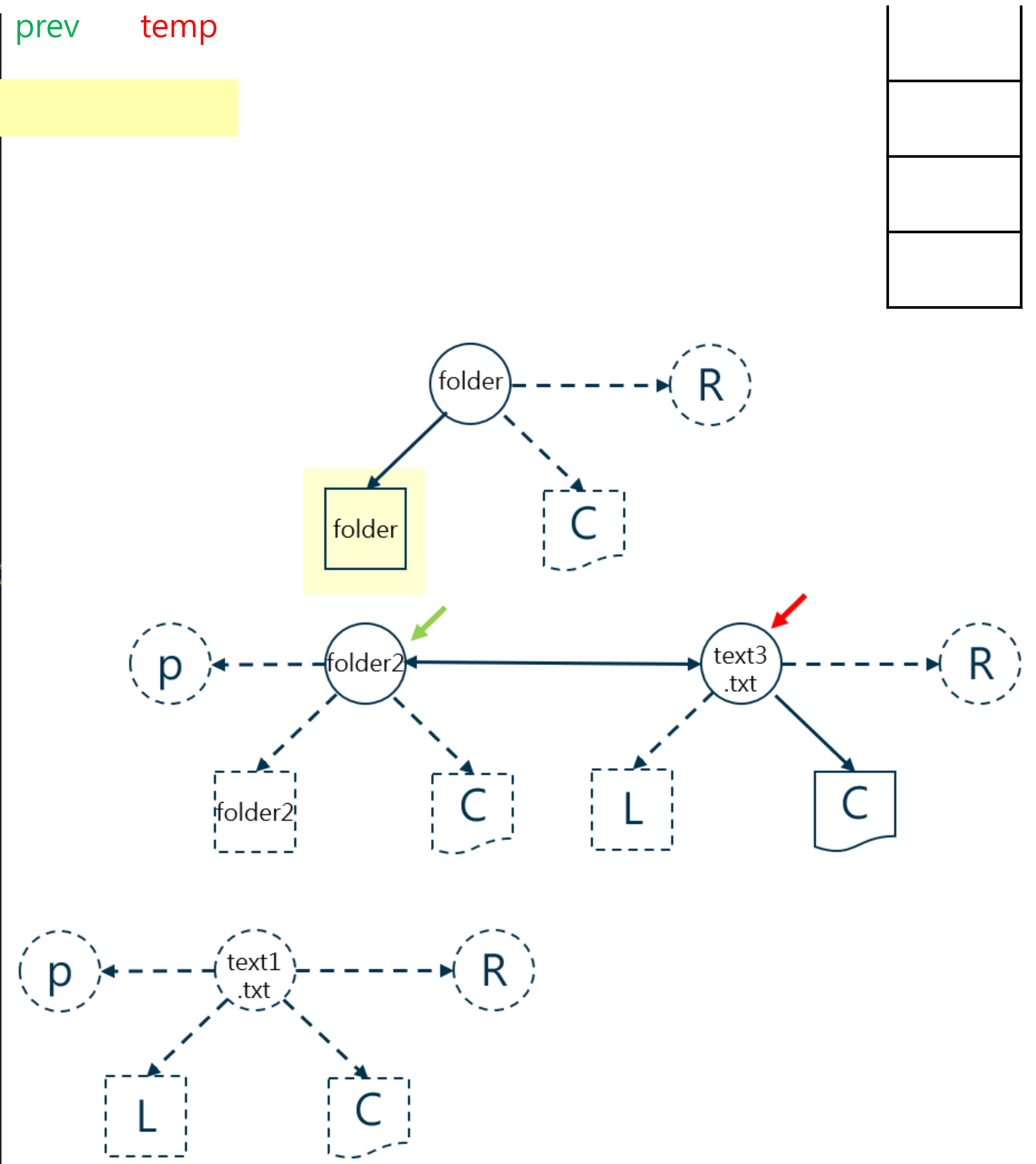


FolderSpaceFree

```c
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

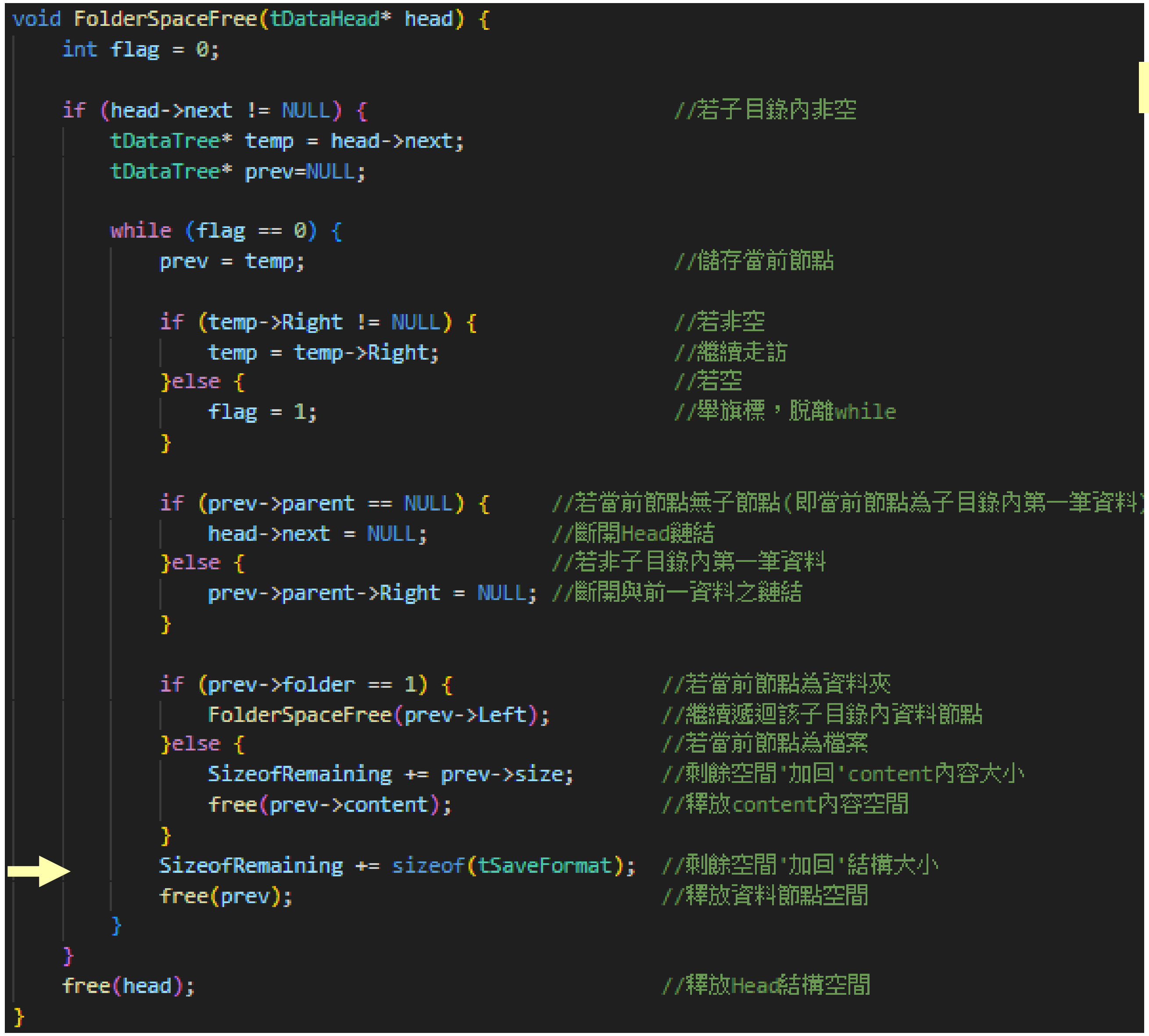


```c
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                  //若當前節點為資料夾
                FolderSpaceFree(prev->Left);          //繼續遞迴該子目錄內資料節點
            }else {                                   //若當前節點為檔案
                SizeofRemaining += prev->size;        //剩餘空間'加回'content內容大小
                free(prev->content);                  //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);   //剩餘空間'加回'結構大小
            free(prev);                               //釋放資料節點空間
        }
    }
    free(head);                                       //釋放Head結構空間
}
```

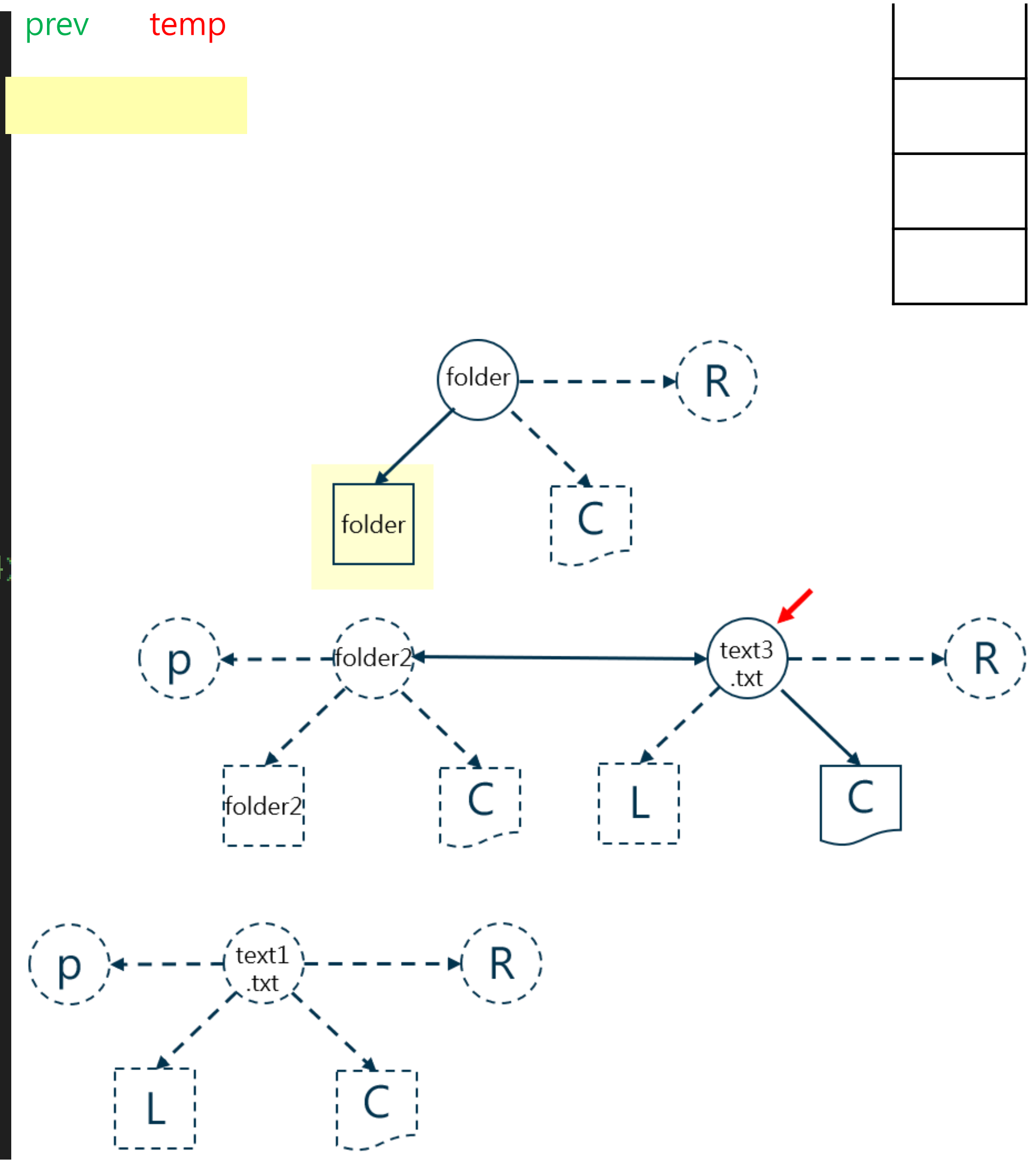


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

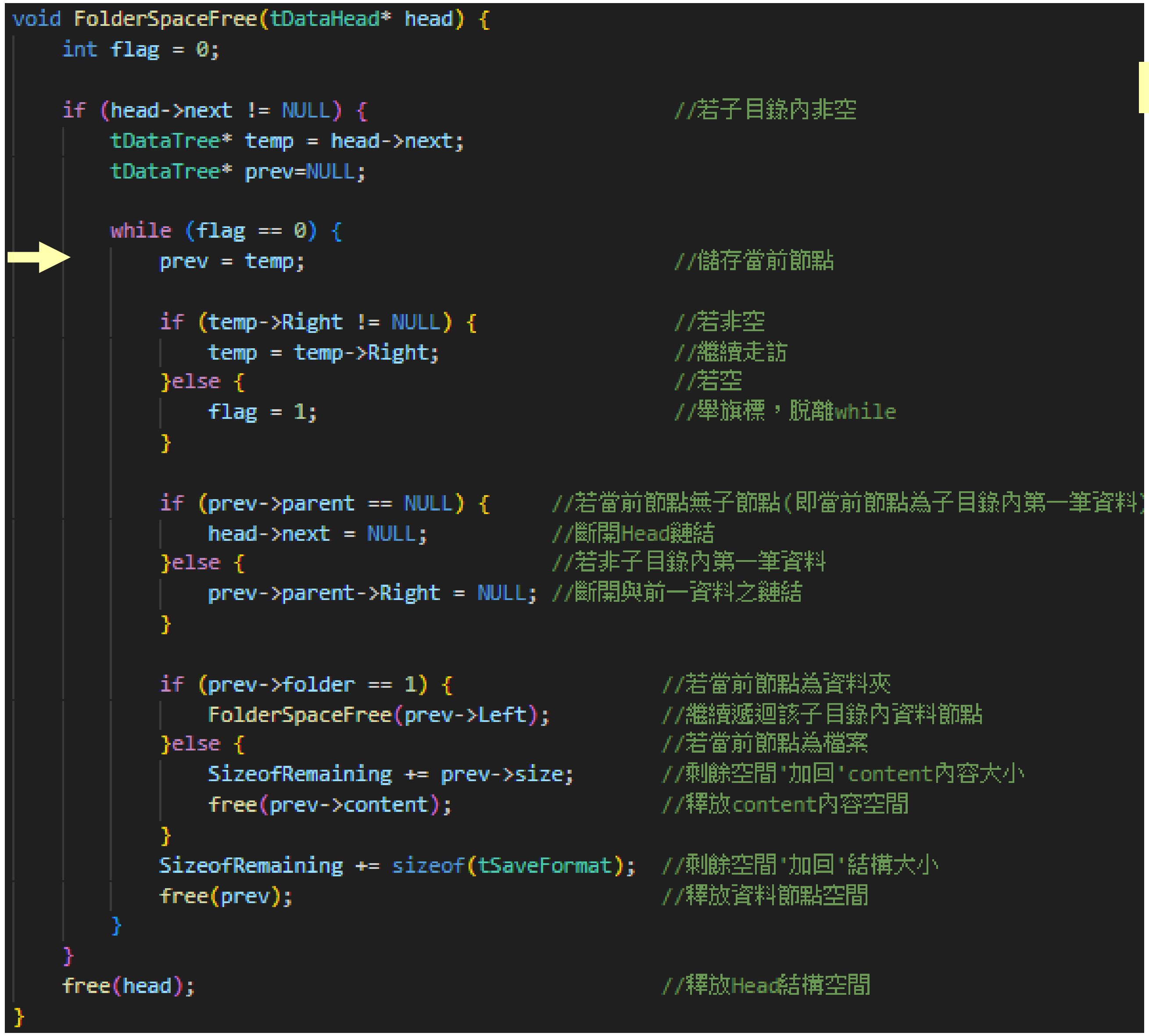


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                           //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                                //儲存當前節點

            if (temp->Right != NULL) {                  //若非空
                temp = temp->Right;                     //繼續走訪
            }else {                                     //若空
                flag = 1;                               //舉旗標，脫離while
            }

            if (prev->parent == NULL) {         //若當前節點無子節點(即當前節點為子目錄內第一筆資料
                head->next = NULL;              //斷開Head鏈結
            }else {                             //若非子目錄內第一筆資料
                prev->parent->Right = NULL; //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                   //若當前節點為資料夾
                FolderSpaceFree(prev->Left);           //繼續遞迴該子目錄內資料節點
            }else {                                    //若當前節點為檔案
                SizeofRemaining += prev->size;         //剩餘空間'加回'content內容大小
                free(prev->content);                   //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);    //剩餘空間'加回'結構大小
            free(prev);                                //釋放資料節點空間
        }
    }
    free(head);                                        //釋放Head結構空間
}
```

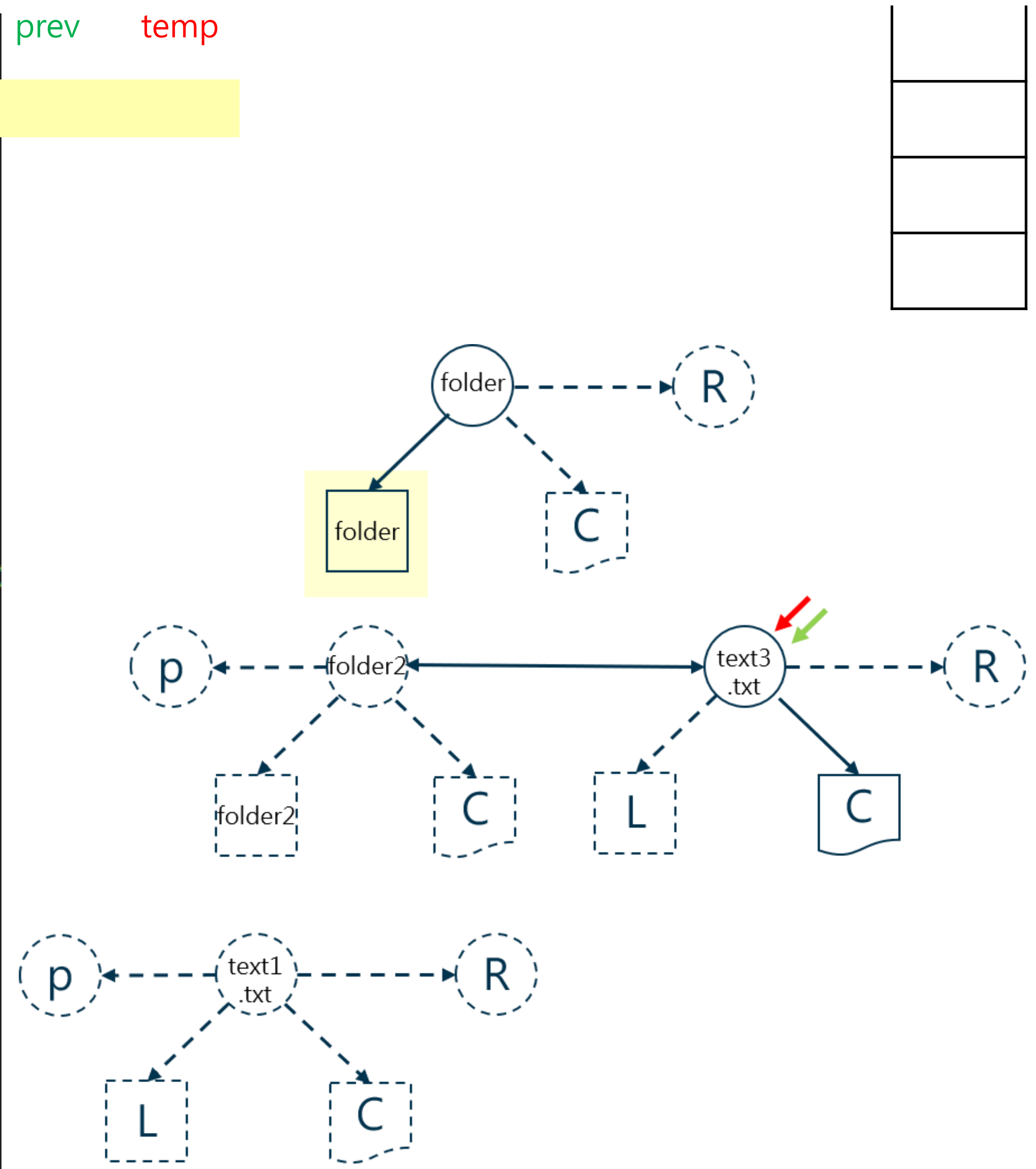


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

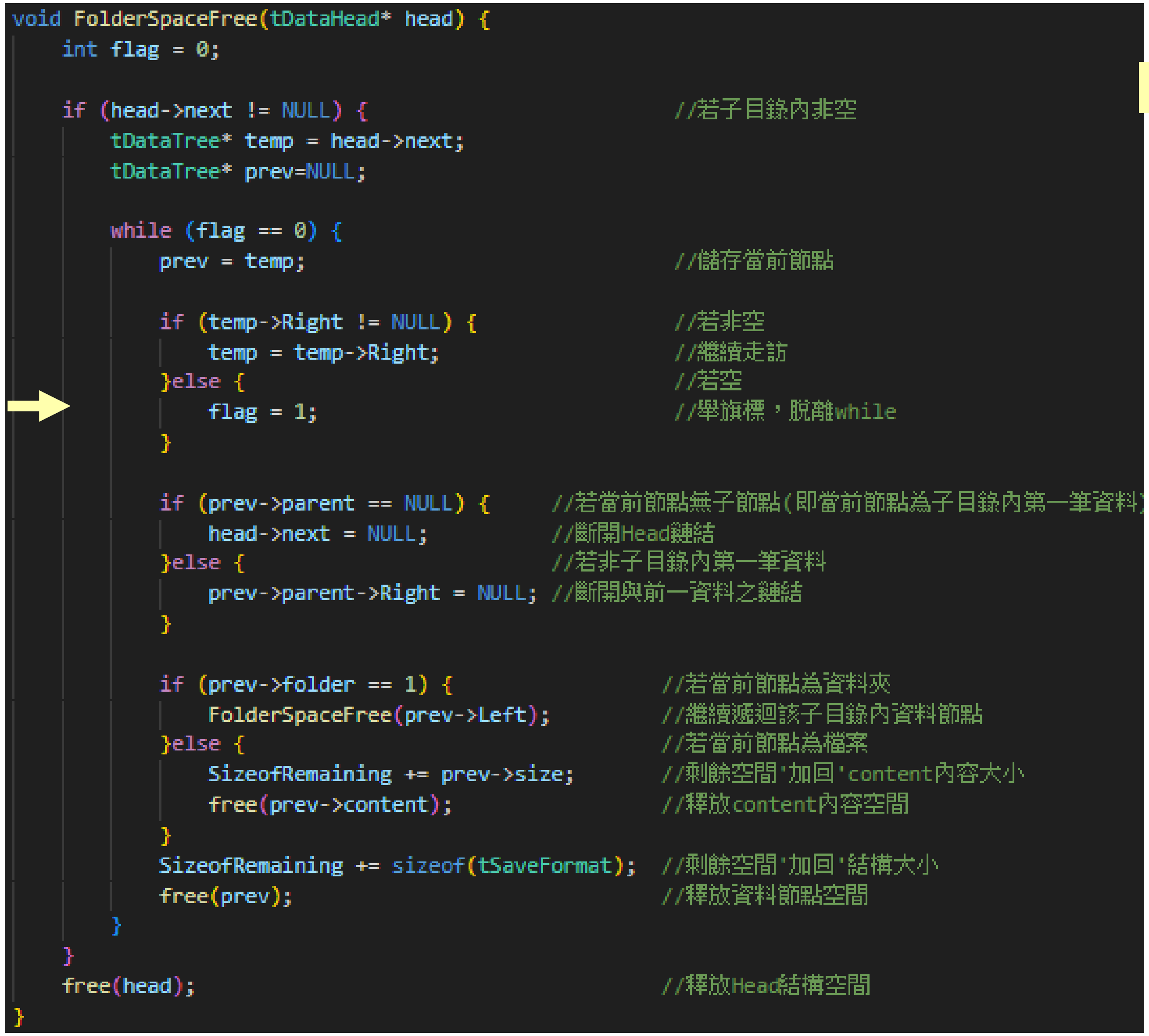


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                 //若當前節點為資料夾
                FolderSpaceFree(prev->Left);         //繼續遞迴該子目錄內資料節點
            }else {                                  //若當前節點為檔案
                SizeofRemaining += prev->size;       //剩餘空間'加回'content內容大小
                free(prev->content);                 //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);  //剩餘空間'加回'結構大小
            free(prev);                              //釋放資料節點空間
        }
    }
    free(head);                                      //釋放Head結構空間
}
```

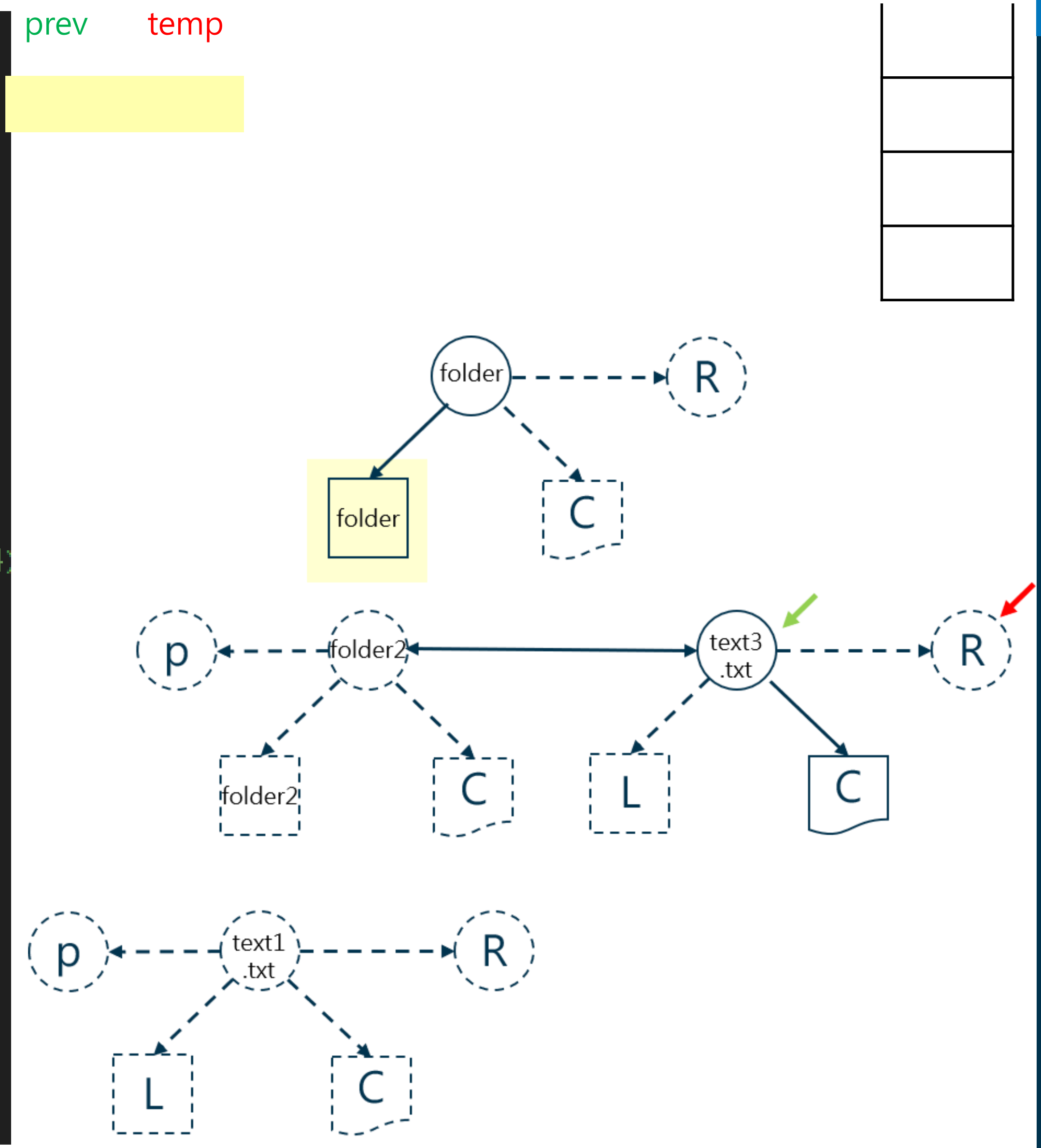


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

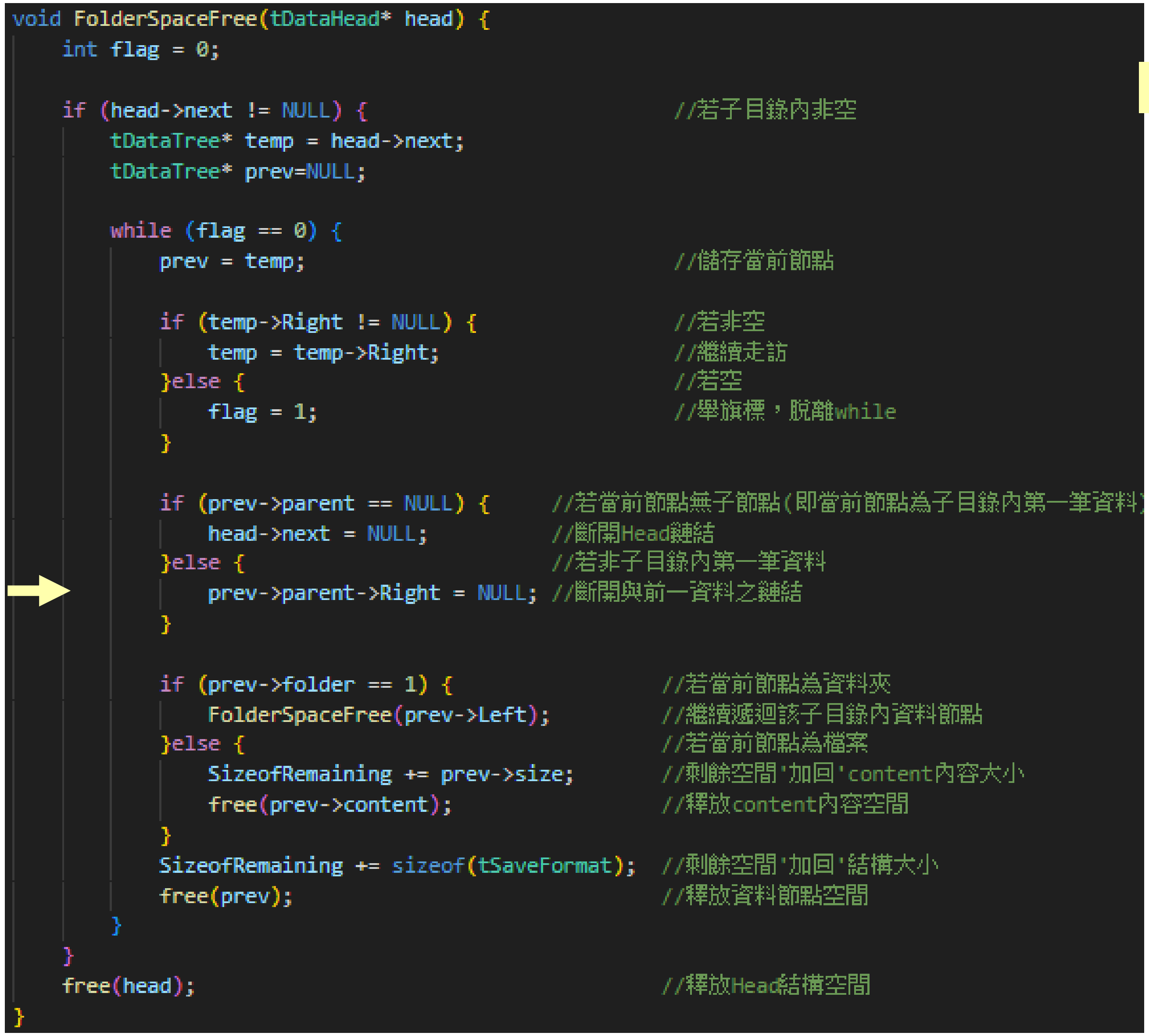


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                           //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                                //儲存當前節點

            if (temp->Right != NULL) {                  //若非空
                temp = temp->Right;                     //繼續走訪
            }else {                                     //若空
                flag = 1;                               //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                   //若當前節點為資料夾
                FolderSpaceFree(prev->Left);           //繼續遞迴該子目錄內資料節點
            }else {                                    //若當前節點為檔案
                SizeofRemaining += prev->size;         //剩餘空間'加回'content內容大小
                free(prev->content);                   //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);    //剩餘空間'加回'結構大小
            free(prev);                                //釋放資料節點空間
        }
    }
    free(head);                                        //釋放Head結構空間
}
```

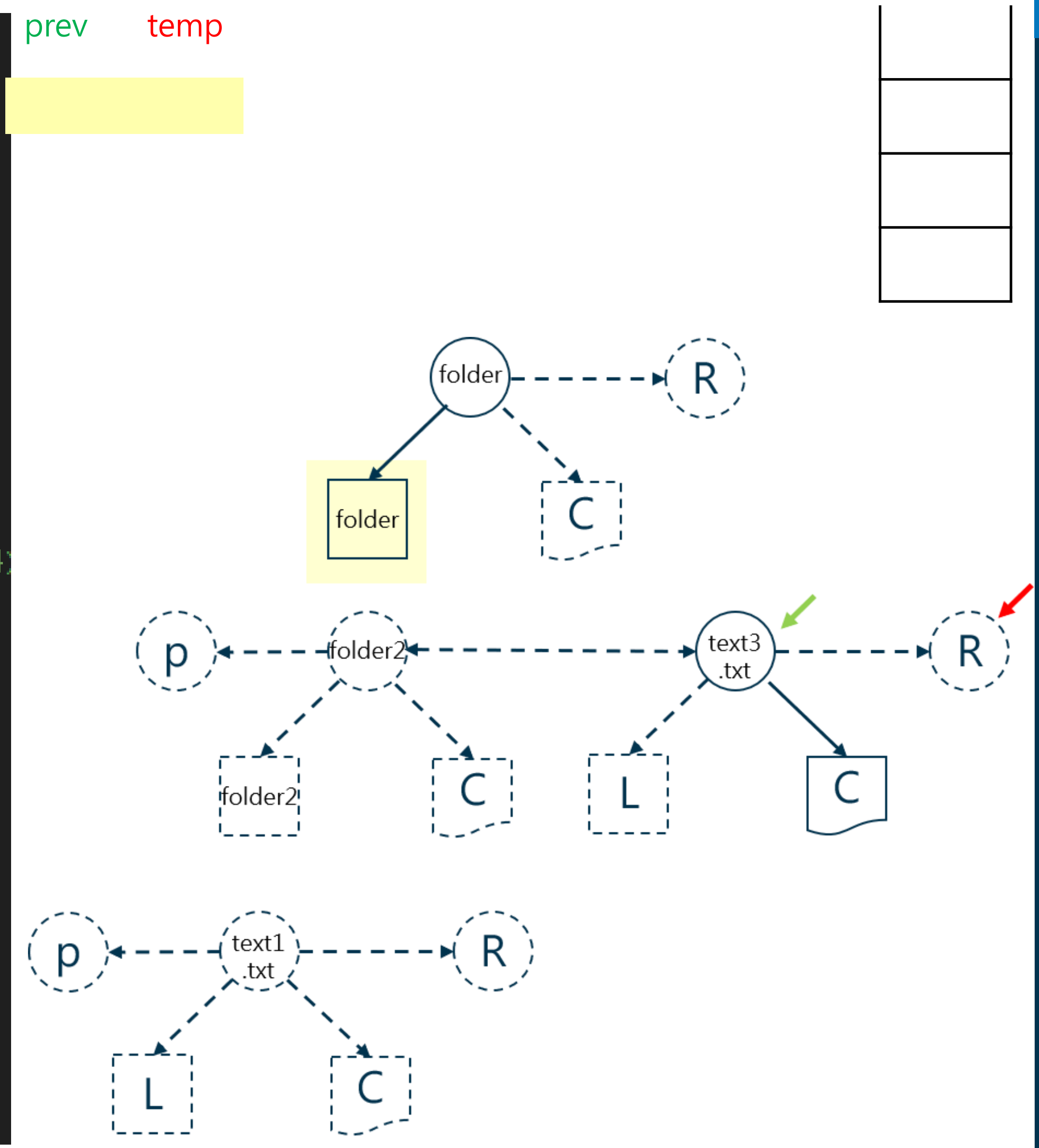


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

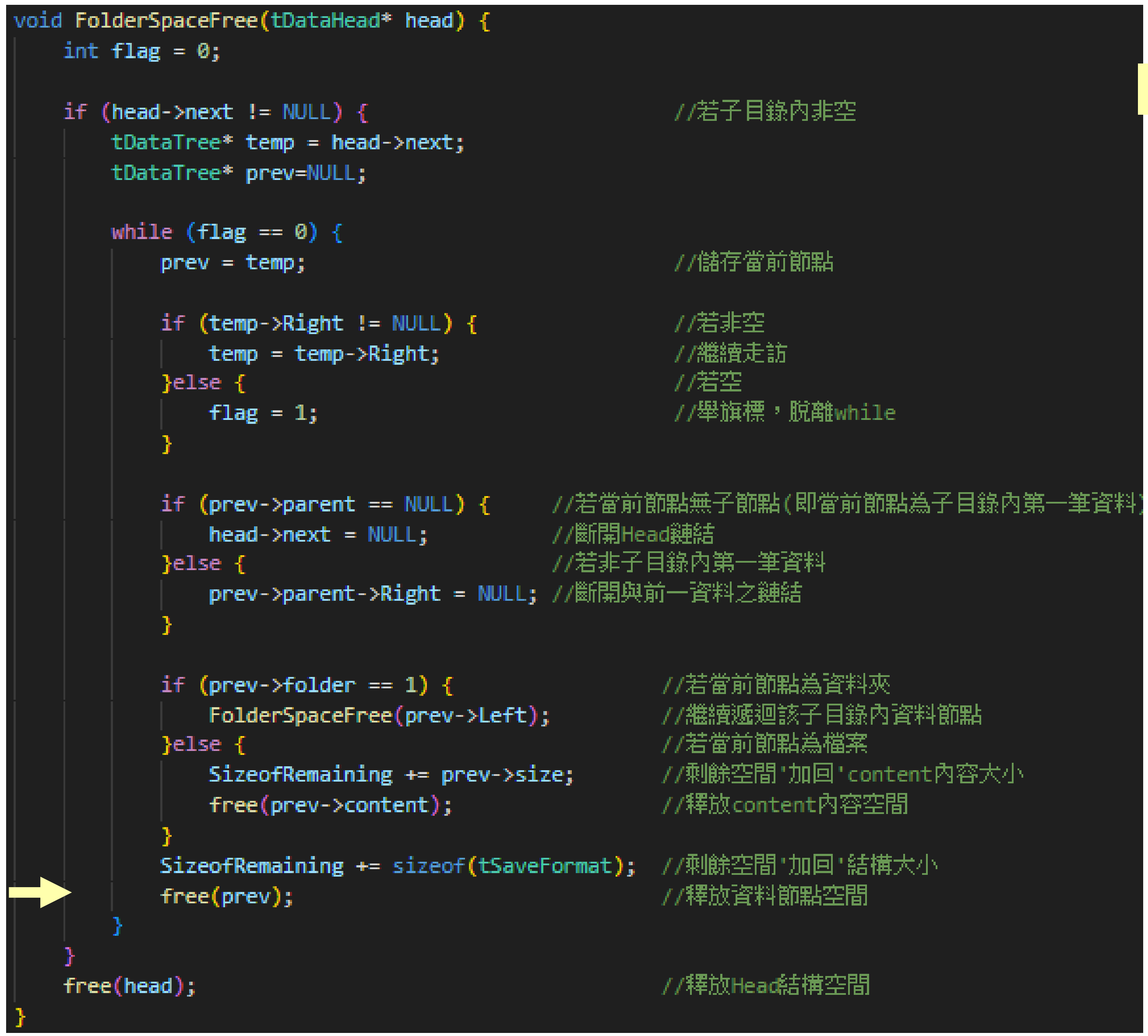


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                  //若當前節點為資料夾
                FolderSpaceFree(prev->Left);          //繼續遞迴該子目錄內資料節點
            }else {                                   //若當前節點為檔案
                SizeofRemaining += prev->size;        //剩餘空間'加回'content內容大小
                free(prev->content);                  //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);   //剩餘空間'加回'結構大小
            free(prev);                               //釋放資料節點空間
        }
    }
    free(head);                                       //釋放Head結構空間
}
```

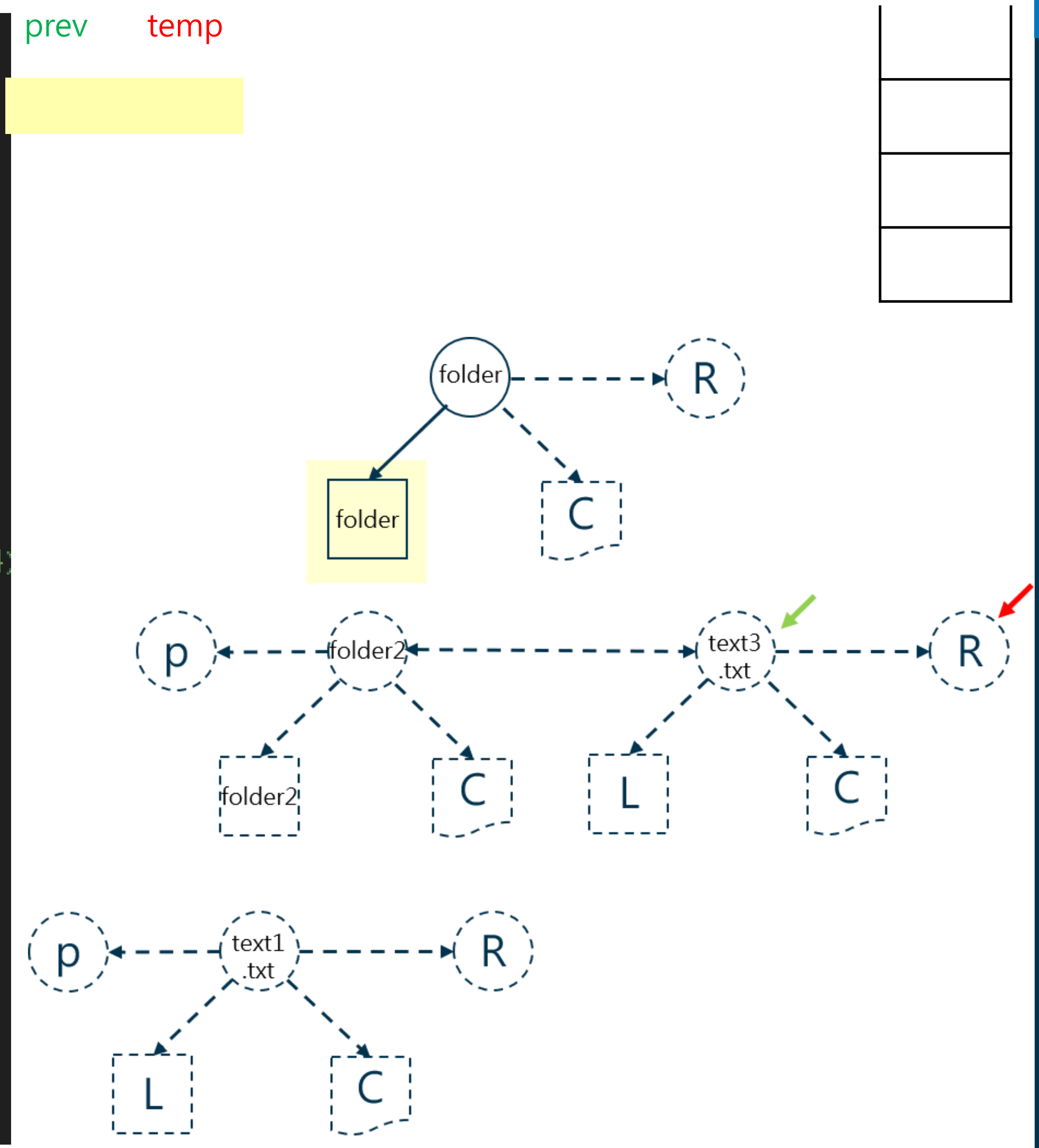


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

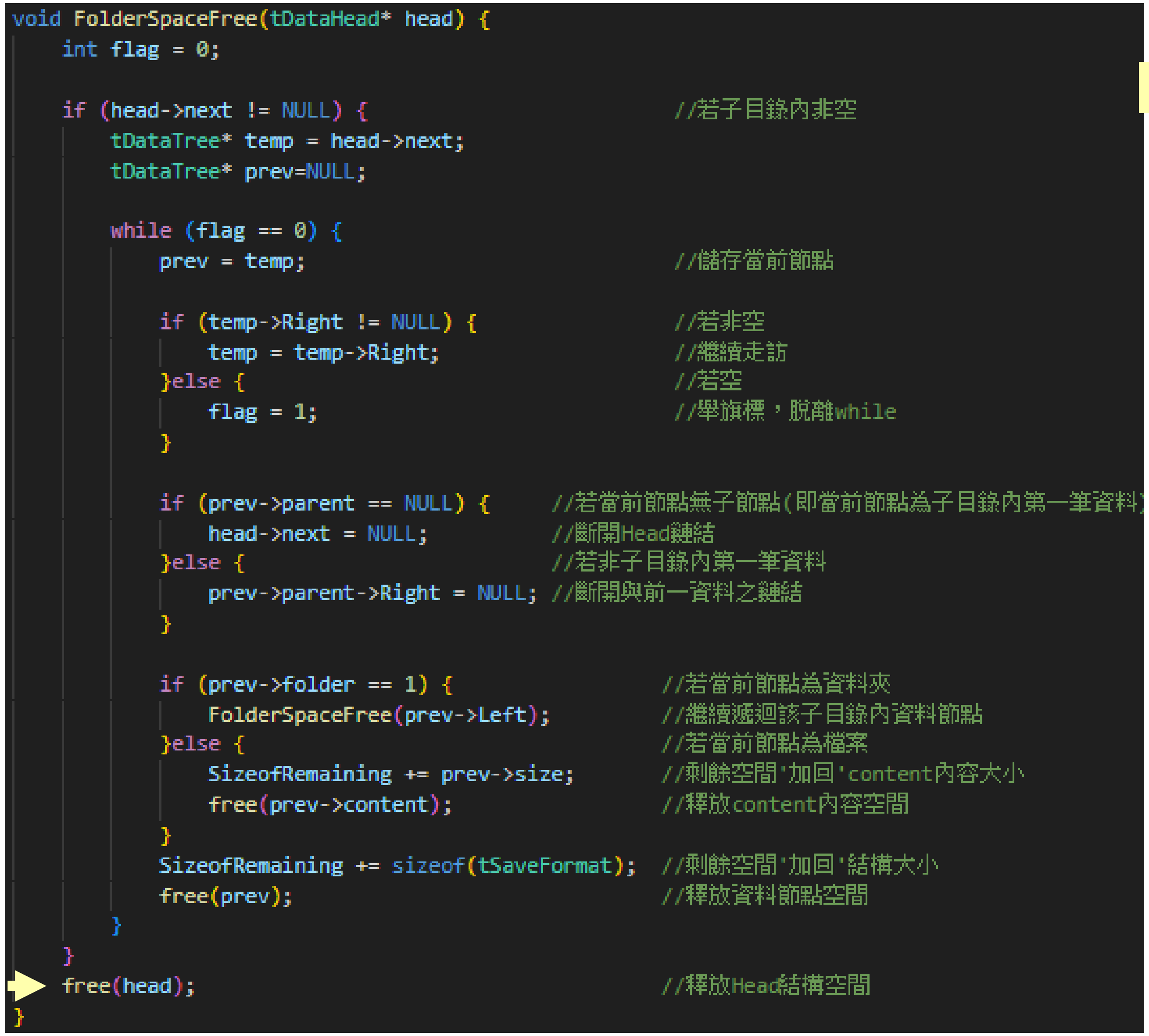


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料)
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                  //若當前節點為資料夾
                FolderSpaceFree(prev->Left);          //繼續遞迴該子目錄內資料節點
            }else {                                   //若當前節點為檔案
                SizeofRemaining += prev->size;        //剩餘空間'加回'content內容大小
                free(prev->content);                  //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);   //剩餘空間'加回'結構大小
            free(prev);                               //釋放資料節點空間
        }
    }
    free(head);                                       //釋放Head結構空間
}
```

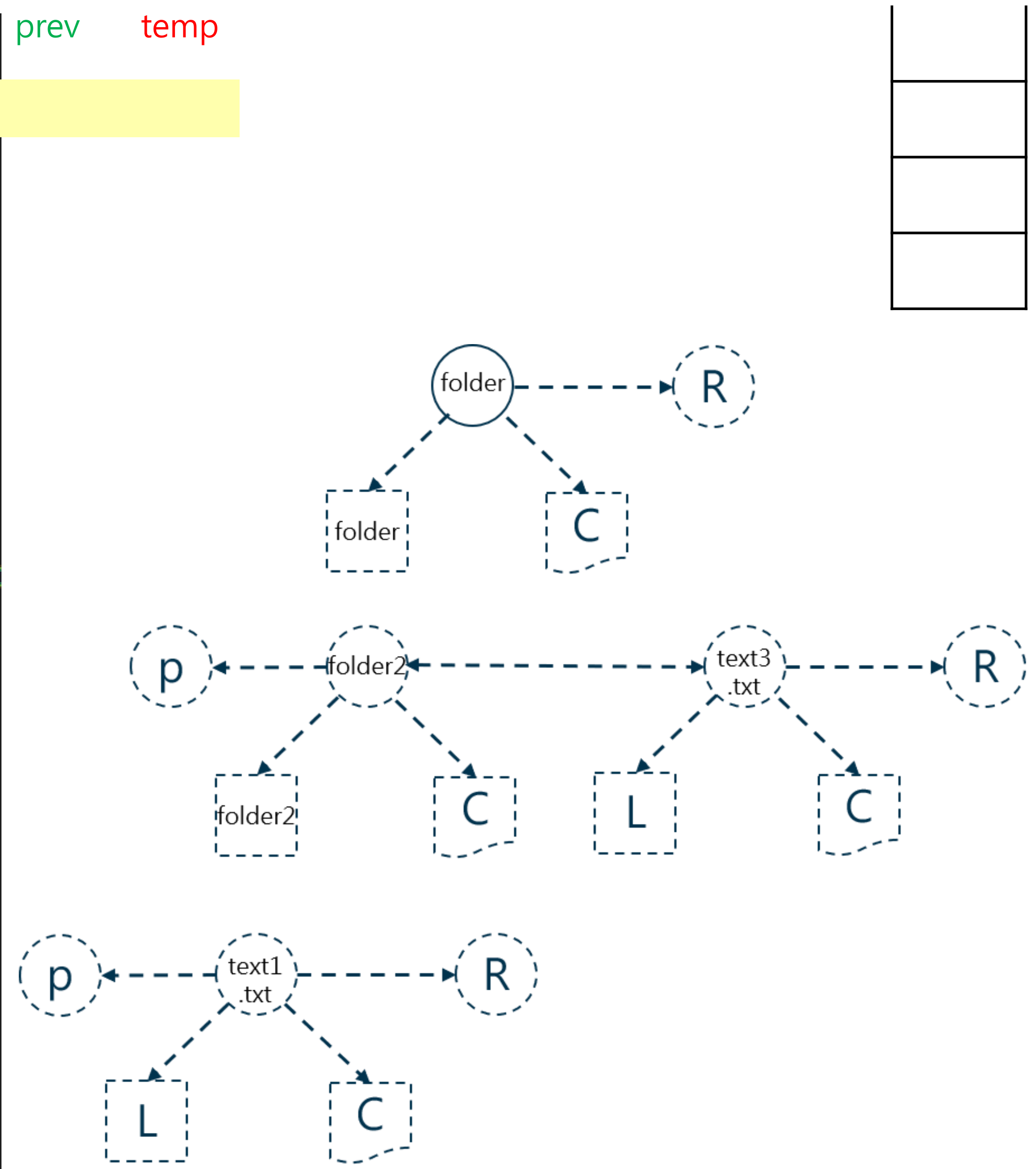


FolderSpaceFree

```
void FolderSpaceFree(tDataHead *head);
FolderSpaceFree(temp->Left);
```

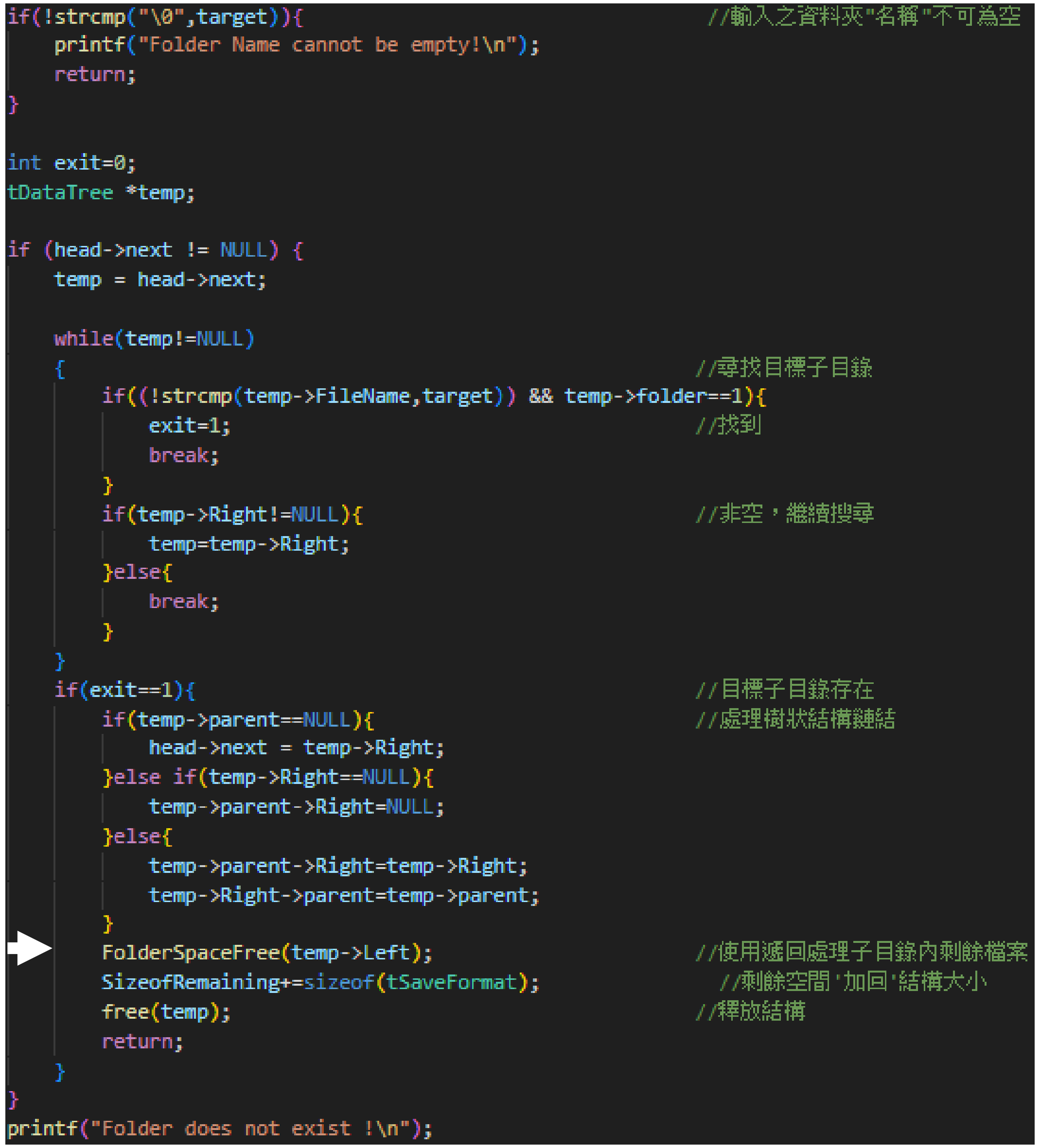


```
void FolderSpaceFree(tDataHead* head) {
    int flag = 0;

    if (head->next != NULL) {                          //若子目錄內非空
        tDataTree* temp = head->next;
        tDataTree* prev=NULL;

        while (flag == 0) {
            prev = temp;                               //儲存當前節點

            if (temp->Right != NULL) {                 //若非空
                temp = temp->Right;                    //繼續走訪
            }else {                                    //若空
                flag = 1;                              //舉旗標，脫離while
            }

            if (prev->parent == NULL) {       //若當前節點無子節點(即當前節點為子目錄內第一筆資料
                head->next = NULL;            //斷開Head鏈結
            }else {                           //若非子目錄內第一筆資料
                prev->parent->Right = NULL;   //斷開與前一資料之鏈結
            }

            if (prev->folder == 1) {                 //若當前節點為資料夾
                FolderSpaceFree(prev->Left);         //繼續遞迴該子目錄內資料節點
            }else {                                  //若當前節點為檔案
                SizeofRemaining += prev->size;       //剩餘空間'加回'content內容大小
                free(prev->content);                 //釋放content內容空間
            }
            SizeofRemaining += sizeof(tSaveFormat);  //剩餘空間'加回'結構大小
            free(prev);                              //釋放資料節點空間
        }
    }
    free(head);                                      //釋放Head結構空間
}
```

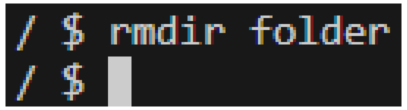


OPER_rmdir

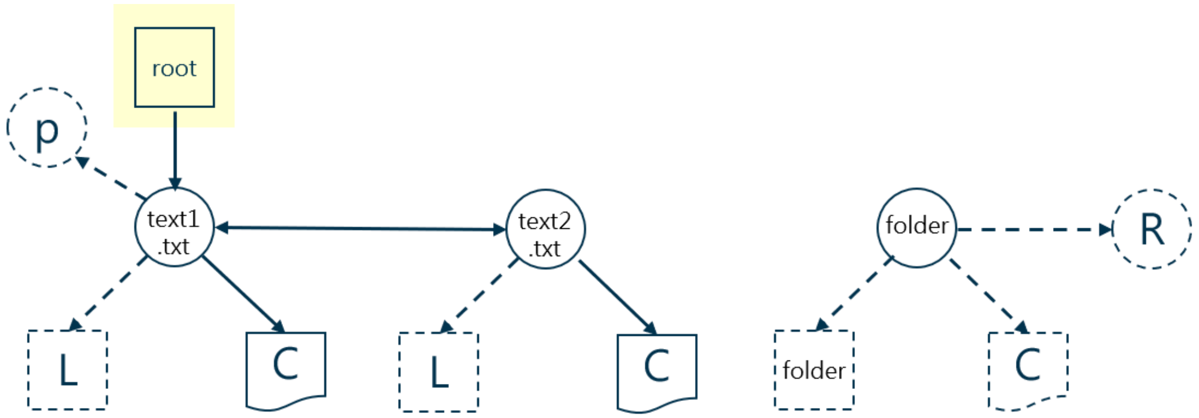
```
void OPER_rmdir(tDataHead *head,char target[]);
OPER_rmdir(curr_Path->Head,oper[1]);
```



```
if(!strcmp("\0",target)){                                          //輸入之資料夾"名稱"不可為空
    printf("Folder Name cannot be empty!\n");
    return;
}

int exit=0;
tDataTree *temp;

if (head->next != NULL) {
    temp = head->next;

    while(temp!=NULL)
    {                                                             //尋找目標子目錄
        if((!strcmp(temp->FileName,target)) && temp->folder==1){
            exit=1;                                               //找到
            break;
        }
        if(temp->Right!=NULL){                                    //非空，繼續搜尋
            temp=temp->Right;
        }else{
            break;
        }
    }
    if(exit==1){                                                  //目標子目錄存在
        if(temp->parent==NULL){                                   //處理樹狀結構鏈結
            head->next = temp->Right;
        }else if(temp->Right==NULL){
            temp->parent->Right=NULL;
        }else{
            temp->parent->Right=temp->Right;
            temp->Right->parent=temp->parent;
        }
        FolderSpaceFree(temp->Left);                              //使用遞回處理子目錄內剩餘檔案
        SizeofRemaining+=sizeof(tSaveFormat);                       //剩餘空間'加回'結構大小
        free(temp);                                               //釋放結構
        return;
    }
}
printf("Folder does not exist !\n");
```

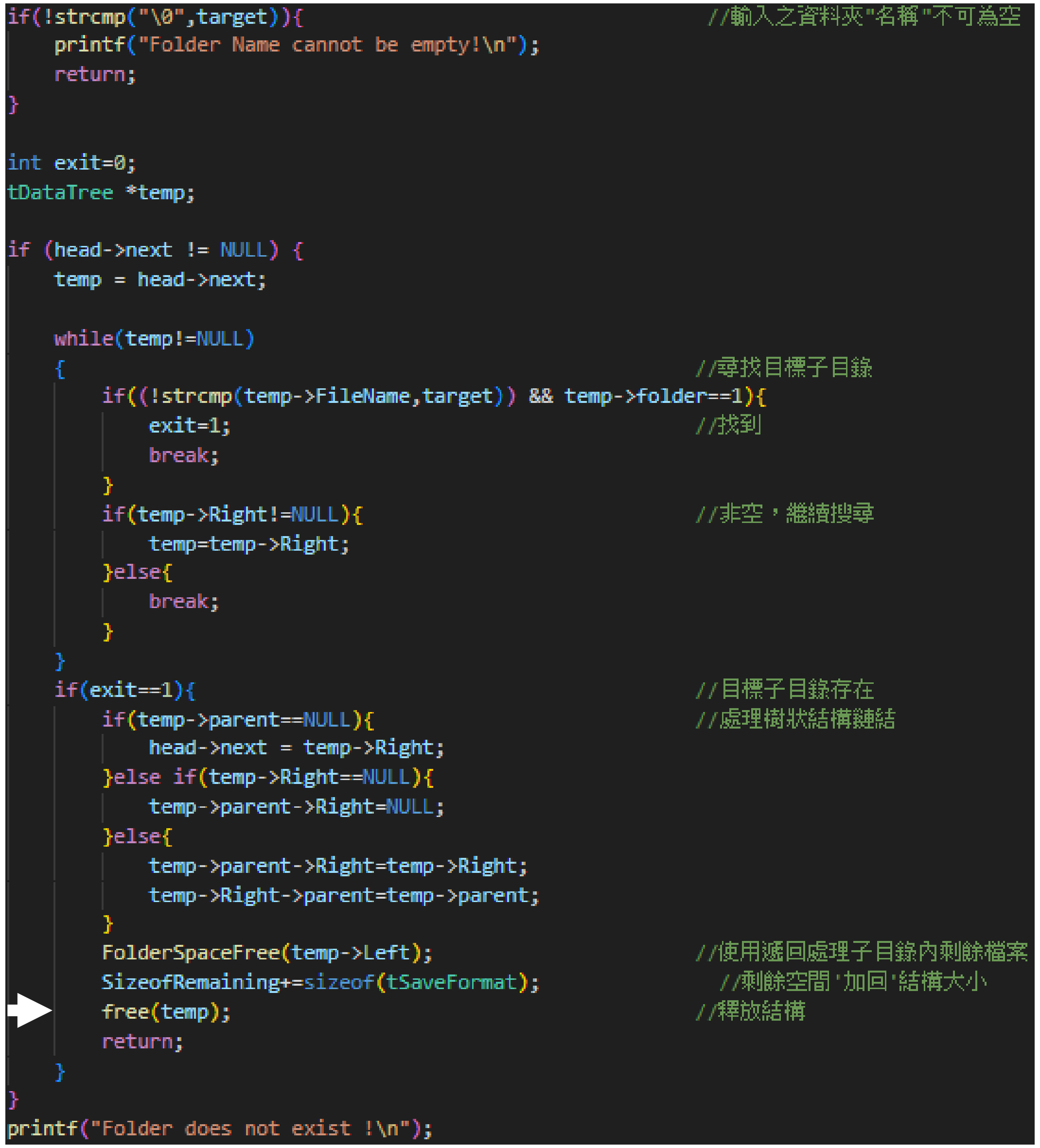


```
/ $ rmdir folder
/ $ 
```



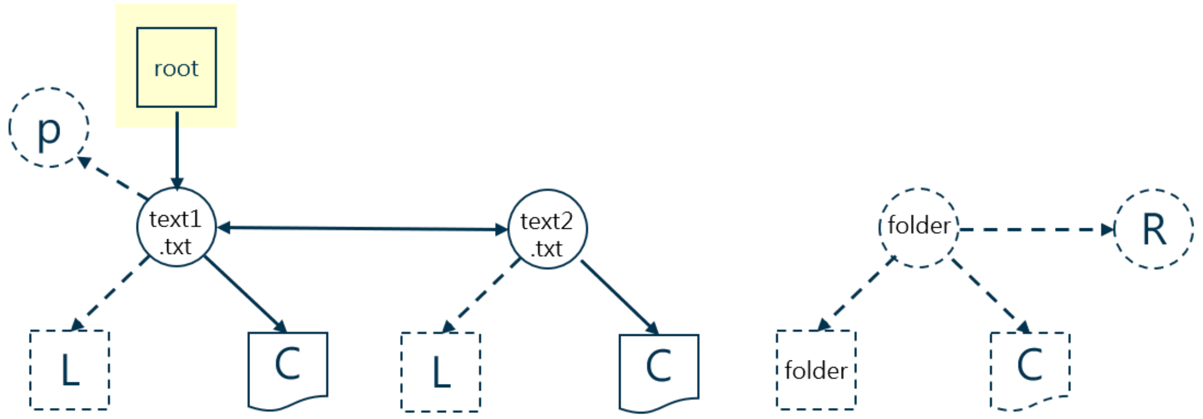
OPER_rmdir

```
void OPER_rmdir(tDataHead *head,char target[]);
OPER_rmdir(curr_Path->Head,oper[1]);
```



```
if(!strcmp("\0",target)){                                     //輸入之資料夾"名稱"不可為空
    printf("Folder Name cannot be empty!\n");
    return;
}

int exit=0;
tDataTree *temp;

if (head->next != NULL) {
    temp = head->next;

    while(temp!=NULL)
    {                                                         //尋找目標子目錄
        if((!strcmp(temp->FileName,target)) && temp->folder==1){
            exit=1;                                           //找到
            break;
        }
        if(temp->Right!=NULL){                                //非空，繼續搜尋
            temp=temp->Right;
        }else{
            break;
        }
    }
    if(exit==1){                                              //目標子目錄存在
        if(temp->parent==NULL){                               //處理樹狀結構鏈結
            head->next = temp->Right;
        }else if(temp->Right==NULL){
            temp->parent->Right=NULL;
        }else{
            temp->parent->Right=temp->Right;
            temp->Right->parent=temp->parent;
        }
        FolderSpaceFree(temp->Left);                          //使用遞回處理子目錄內剩餘檔案
        SizeofRemaining+=sizeof(tSaveFormat);                   //剩餘空間'加回'結構大小
        free(temp);                                           //釋放結構
        return;
    }
}
printf("Folder does not exist !\n");
```

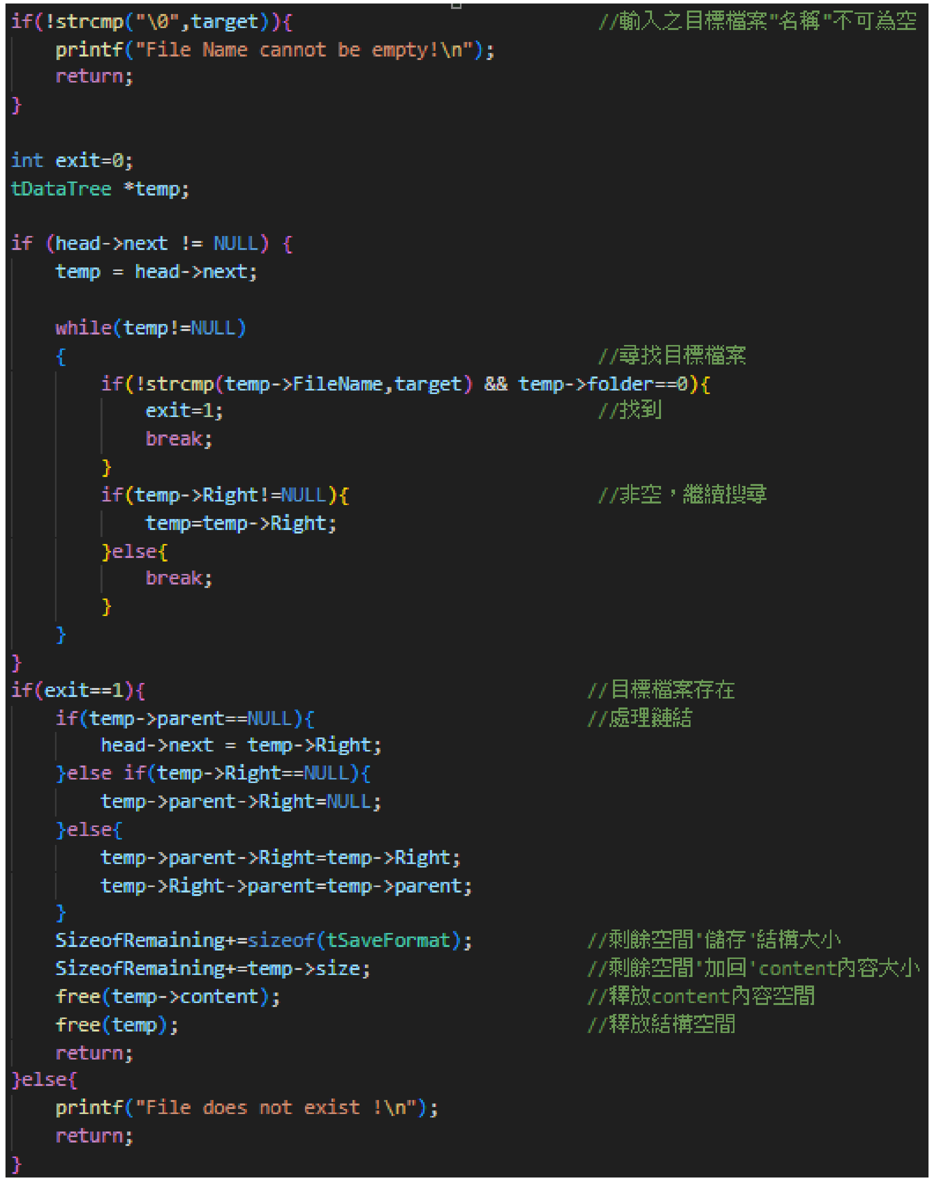
```
/ $ rmdir folder
/ $ 
```



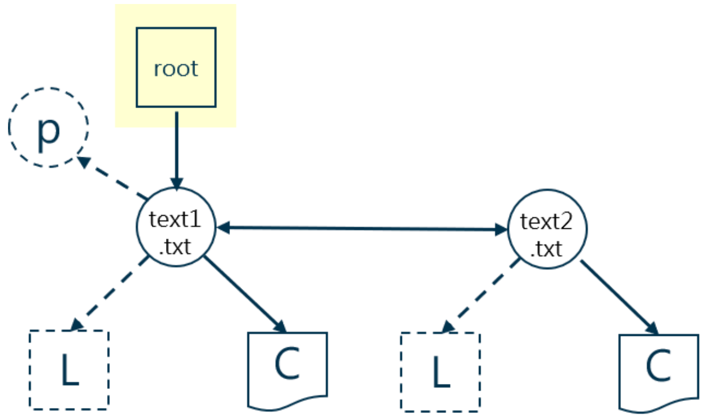
OPER_rm

```
void OPER_rm(tDataHead *head,char target[]);
```

```
OPER_rm(curr_Path->Head,oper[1]);
```

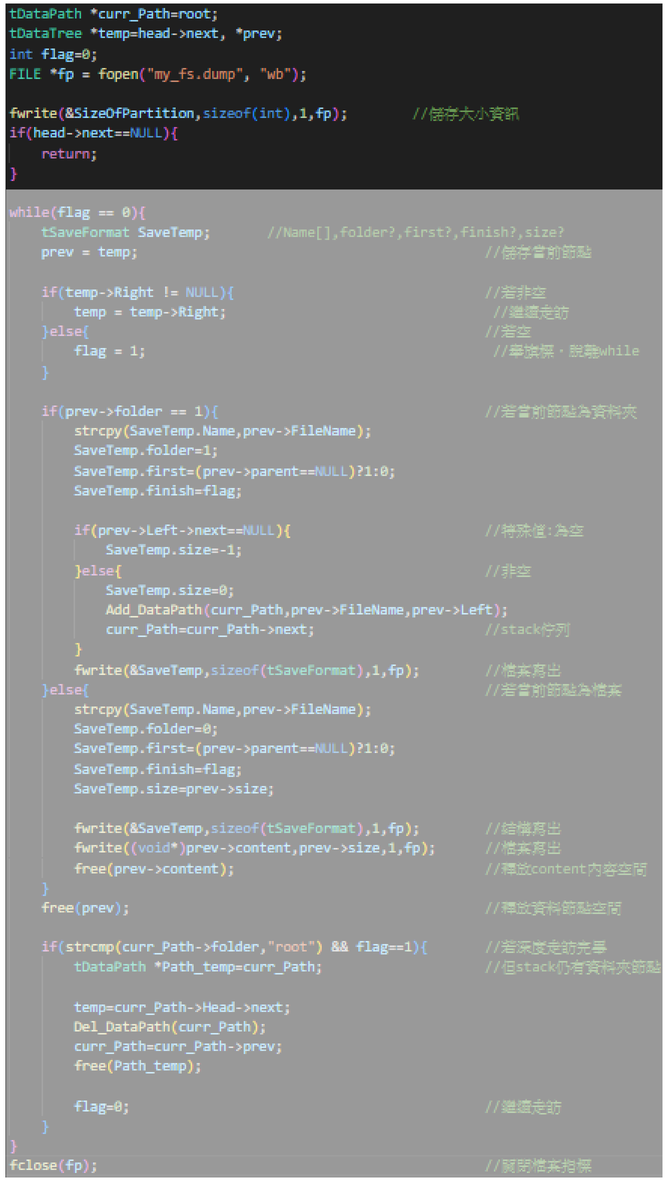


```
if(!strcmp("\0",target)){                             //輸入之目標檔案"名稱"不可為空
    printf("File Name cannot be empty!\n");
    return;
}

int exit=0;
tDataTree *temp;

if (head->next != NULL) {
    temp = head->next;

    while(temp!=NULL)
    {                                                 //尋找目標檔案
        if(!strcmp(temp->FileName,target) && temp->folder==0){
            exit=1;                                   //找到
            break;
        }
        if(temp->Right!=NULL){                        //非空，繼續搜尋
            temp=temp->Right;
        }else{
            break;
        }
    }
}
if(exit==1){                                         //目標檔案存在
    if(temp->parent==NULL){                          //處理鏈結
        head->next = temp->Right;
    }else if(temp->Right==NULL){
        temp->parent->Right=NULL;
    }else{
        temp->parent->Right=temp->Right;
        temp->Right->parent=temp->parent;
    }
    SizeofRemaining+=sizeof(tSaveFormat);            //剩餘空間'儲存'結構大小
    SizeofRemaining+=temp->size;                     //剩餘空間'加回'content內容大小
    free(temp->content);                             //釋放content內容空間
    free(temp);                                      //釋放結構空間
    return;
}else{
    printf("File does not exist !\n");
    return;
}
```

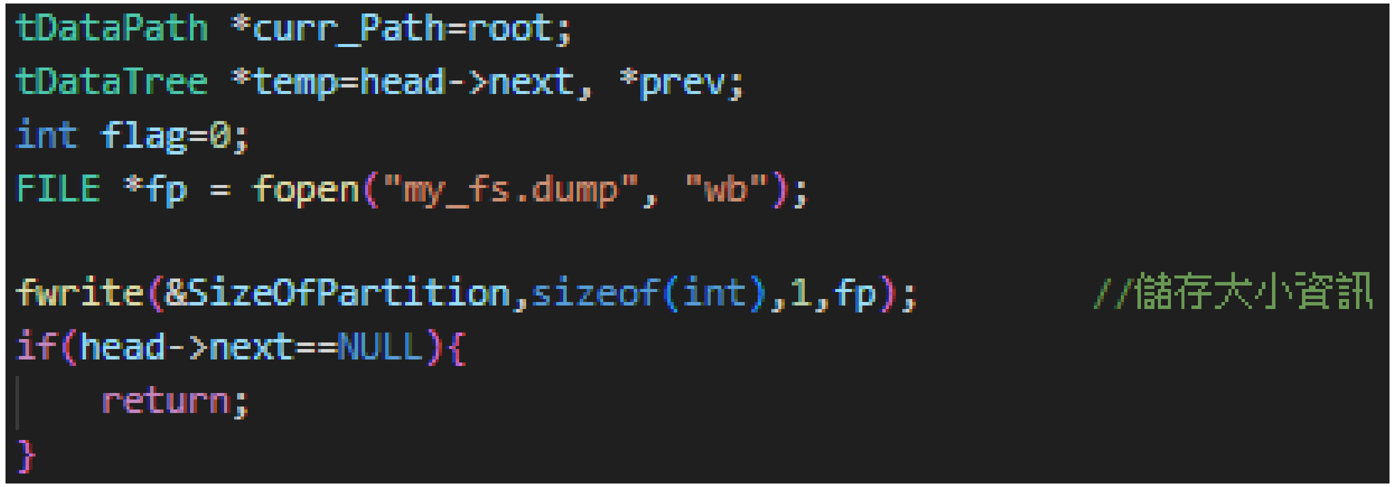


OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```

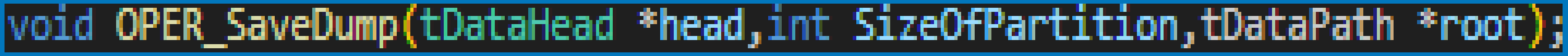


```
tDataPath *curr_Path=root;
tDataTree *temp=head->next, *prev;
int flag=0;
FILE *fp = fopen("my_fs.dump", "wb");

fwrite(&SizeOfPartition,sizeof(int),1,fp);        //儲存大小資訊
if(head->next==NULL){
    return;
}

while(flag == 0){
    tSaveFormat SaveTemp;       //Name[],folder?,first?,finish?,size?
    prev = temp;                                 //儲存當前節點

    if(temp->Right != NULL){                     //若非空
        temp = temp->Right;                       //繼續走訪
    }else{                                       //若空
        flag = 1;                                 //舉旗標，脫離while
    }

    if(prev->folder == 1){                       //若當前節點為資料夾
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=1;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;

        if(prev->Left->next==NULL){              //特殊值:為空
            SaveTemp.size=-1;
        }else{                                   //非空
            SaveTemp.size=0;
            Add_DataPath(curr_Path,prev->FileName,prev->Left);
            curr_Path=curr_Path->next;           //stack佇列
        }
        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);       //檔案寫出
    }else{                                       //若當前節點為檔案
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=0;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;
        SaveTemp.size=prev->size;

        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);       //結構寫出
        fwrite((void*)prev->content,prev->size,1,fp);     //檔案寫出
        free(prev->content);                     //釋放content內容空間
    }
    free(prev);                                  //釋放資料節點空間

    if(strcmp(curr_Path->folder,"root") && flag==1){      //若深度走訪完畢
        tDataPath *Path_temp=curr_Path;          //但stack仍有資料夾節點

        temp=curr_Path->Head->next;
        Del_DataPath(curr_Path);
        curr_Path=curr_Path->prev;
        free(Path_temp);

        flag=0;                                  //繼續走訪
    }
}
fclose(fp);                                      //關閉檔案指標
```

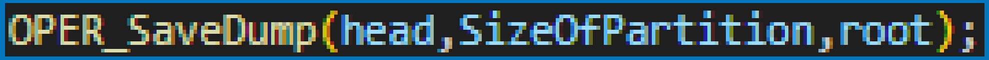


```
tDataPath *curr_Path=root;
tDataTree *temp=head->next, *prev;
int flag=0;
FILE *fp = fopen("my_fs.dump", "wb");

fwrite(&SizeOfPartition,sizeof(int),1,fp);        //儲存大小資訊
if(head->next==NULL){
    return;
}
```

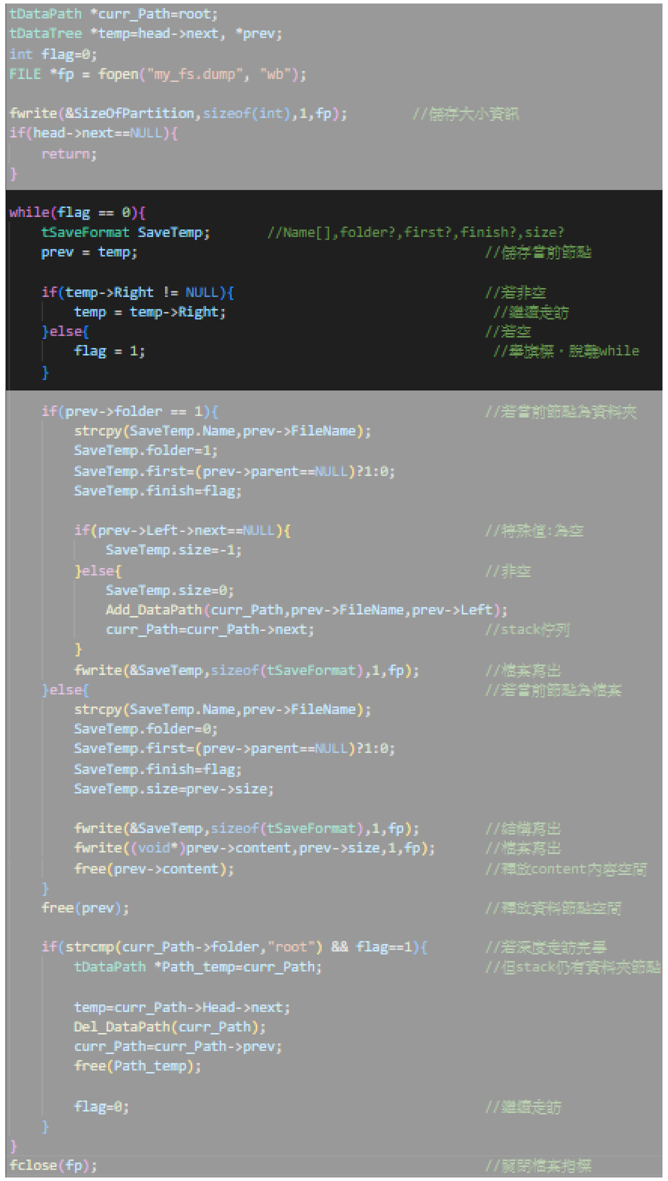
OPER_SaveDump



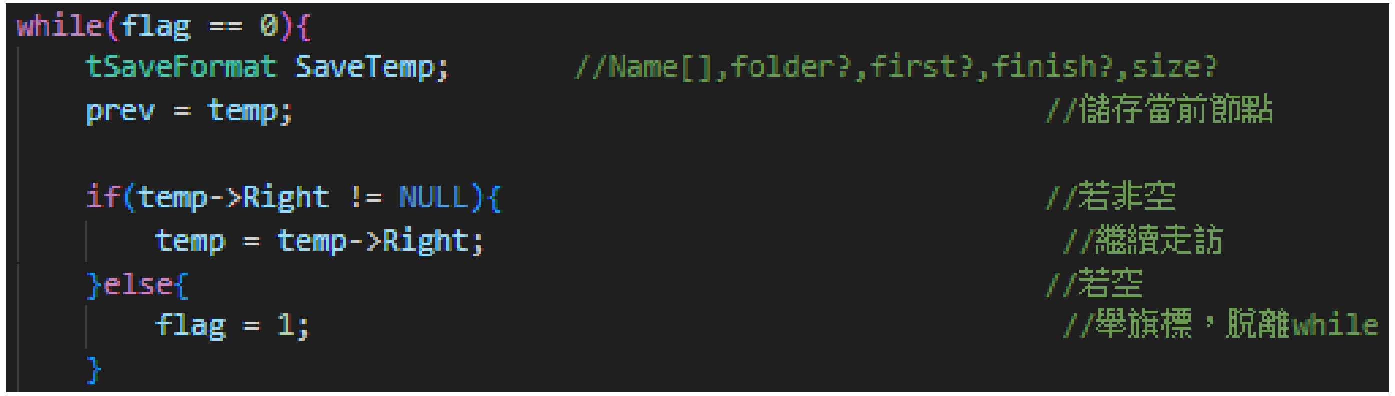
```c
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
```



```c
OPER_SaveDump(head,SizeOfPartition,root);
```



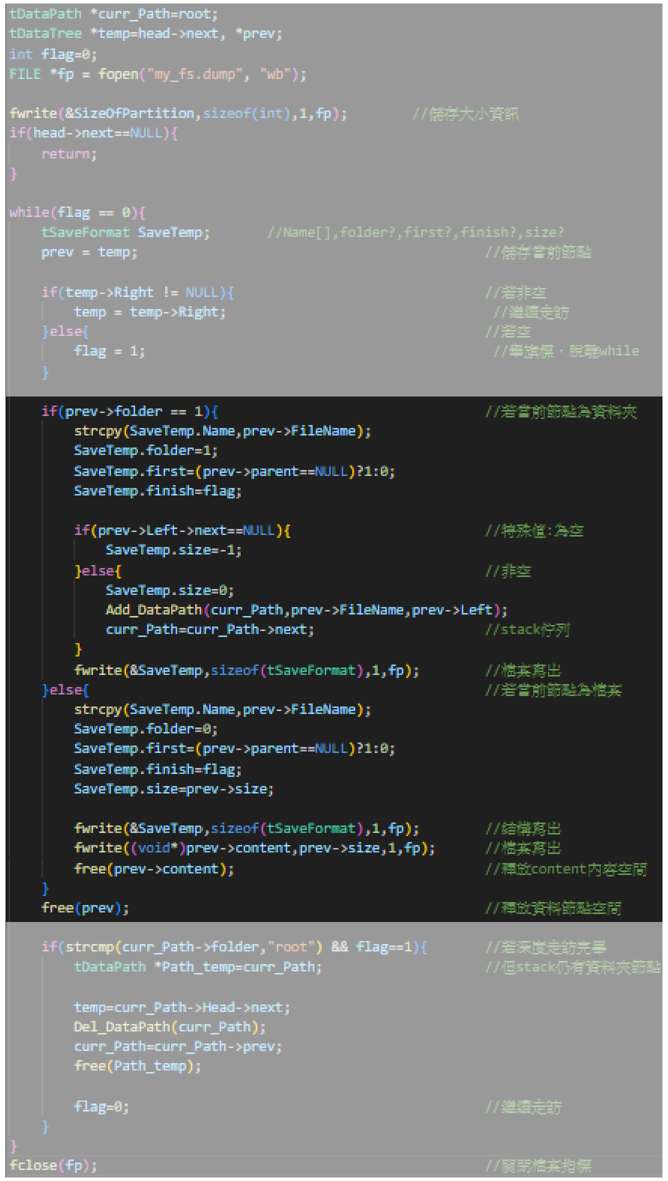
```c
tDataPath *curr_Path=root;
tDataTree *temp=head->next, *prev;
int flag=0;
FILE *fp = fopen("my_fs.dump", "wb");

fwrite(&SizeOfPartition,sizeof(int),1,fp);        //儲存大小資訊
if(head->next==NULL){
    return;
}

while(flag == 0){
    tSaveFormat SaveTemp;       //Name[],folder?,first?,finish?,size?
    prev = temp;                                          //儲存當前節點

    if(temp->Right != NULL){                              //若非空
        temp = temp->Right;                                //繼續走訪
    }else{                                                //若空
        flag = 1;                                          //舉旗標，脫離while
    }

    if(prev->folder == 1){                                //若當前節點為資料夾
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=1;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;

        if(prev->Left->next==NULL){                       //特殊值:為空
            SaveTemp.size=-1;
        }else{                                            //非空
            SaveTemp.size=0;
            Add_DataPath(curr_Path,prev->FileName,prev->Left);
            curr_Path=curr_Path->next;                    //stack佇列
        }
        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);       //檔案寫出
    }else{                                                //若當前節點為檔案
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=0;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;
        SaveTemp.size=prev->size;

        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);       //結構寫出
        fwrite((void*)prev->content,prev->size,1,fp);     //檔案寫出
        free(prev->content);                              //釋放content內容空間
    }
    free(prev);                                           //釋放資料節點空間

    if(strcmp(curr_Path->folder,"root") && flag==1){      //若深度走訪完畢
        tDataPath *Path_temp=curr_Path;                   //但stack仍有資料夾節點

        temp=curr_Path->Head->next;
        Del_DataPath(curr_Path);
        curr_Path=curr_Path->prev;
        free(Path_temp);

        flag=0;                                           //繼續走訪
    }
}
fclose(fp);                                               //關閉檔案指標
```



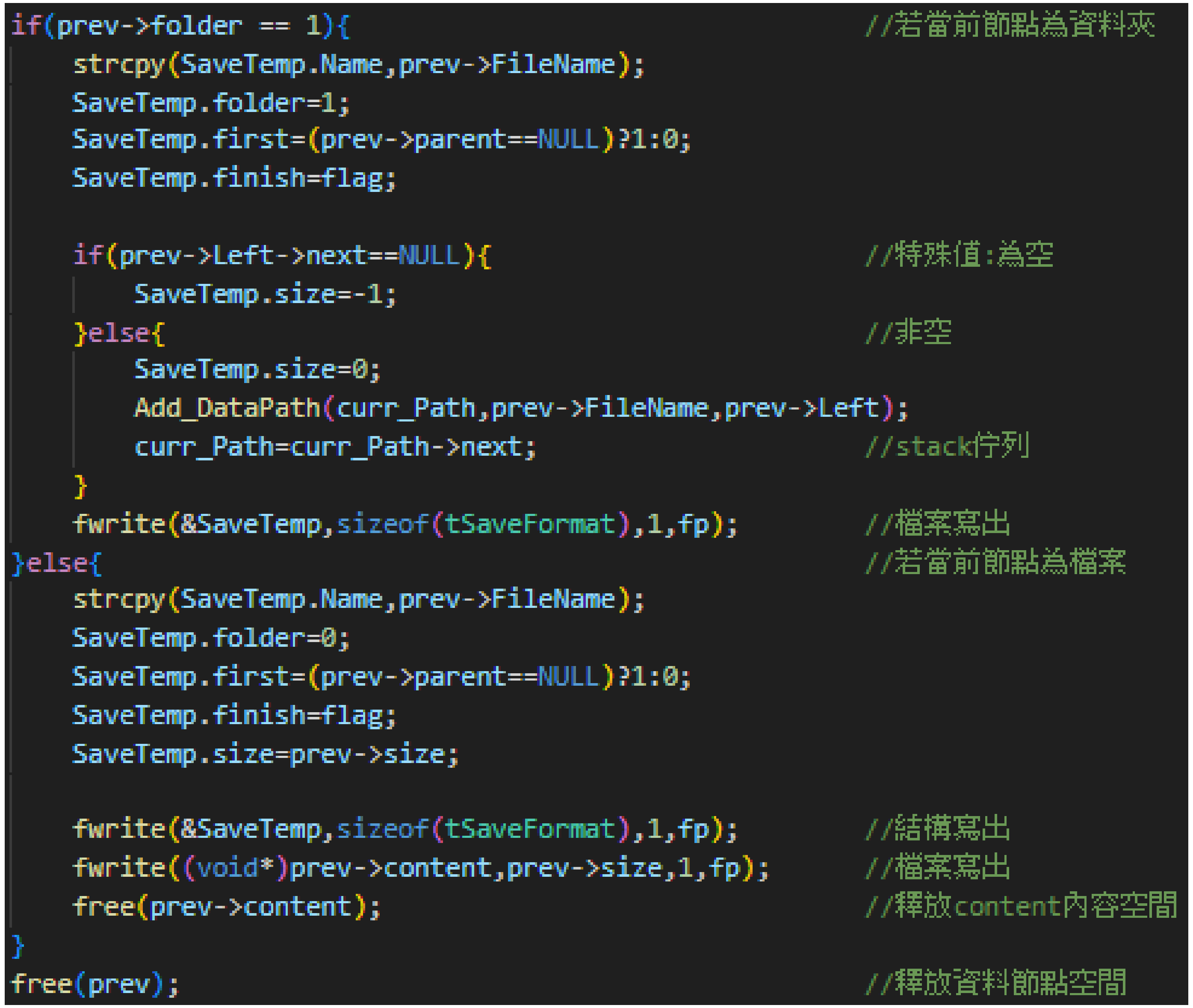
```c
while(flag == 0){
    tSaveFormat SaveTemp;       //Name[],folder?,first?,finish?,size?
    prev = temp;                                          //儲存當前節點

    if(temp->Right != NULL){                              //若非空
        temp = temp->Right;                                //繼續走訪
    }else{                                                //若空
        flag = 1;                                          //舉旗標，脫離while
    }
```

OPER_SaveDump

```c
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



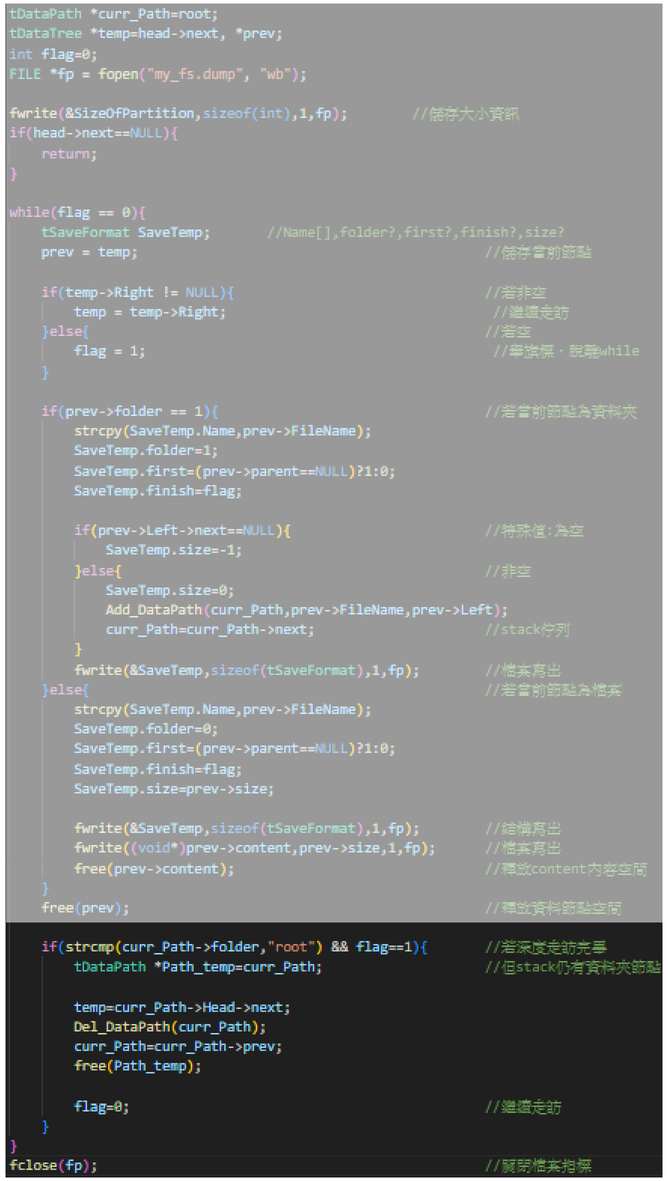
```c
tDataPath *curr_Path=root;
tDataTree *temp=head->next, *prev;
int flag=0;
FILE *fp = fopen("my_fs.dump", "wb");

fwrite(&SizeOfPartition,sizeof(int),1,fp);          //儲存大小資訊
if(head->next==NULL){
    return;
}

while(flag == 0){
    tSaveFormat SaveTemp;       //Name[],folder?,first?,finish?,size?
    prev = temp;                                     //儲存當前節點

    if(temp->Right != NULL){                         //若非空
        temp = temp->Right;                           //繼續走訪
    }else{                                           //若空
        flag = 1;                                     //舉旗標，脫離while
    }

    if(prev->folder == 1){                           //若當前節點為資料夾
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=1;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;

        if(prev->Left->next==NULL){                  //特殊值:為空
            SaveTemp.size=-1;
        }else{                                       //非空
            SaveTemp.size=0;
            Add_DataPath(curr_Path,prev->FileName,prev->Left);
            curr_Path=curr_Path->next;               //stack佇列
        }
        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);  //檔案寫出
    }else{                                           //若當前節點為檔案
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=0;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;
        SaveTemp.size=prev->size;

        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);  //結構寫出
        fwrite((void*)prev->content,prev->size,1,fp); //檔案寫出
        free(prev->content);                         //釋放content內容空間
    }
    free(prev);                                      //釋放資料節點空間

    if(strcmp(curr_Path->folder,"root") && flag==1){ //若深度走訪完畢
        tDataPath *Path_temp=curr_Path;              //但stack仍有資料夾節點

        temp=curr_Path->Head->next;
        Del_DataPath(curr_Path);
        curr_Path=curr_Path->prev;
        free(Path_temp);

        flag=0;                                      //繼續走訪
    }
}
fclose(fp);                                          //關閉檔案指標
```



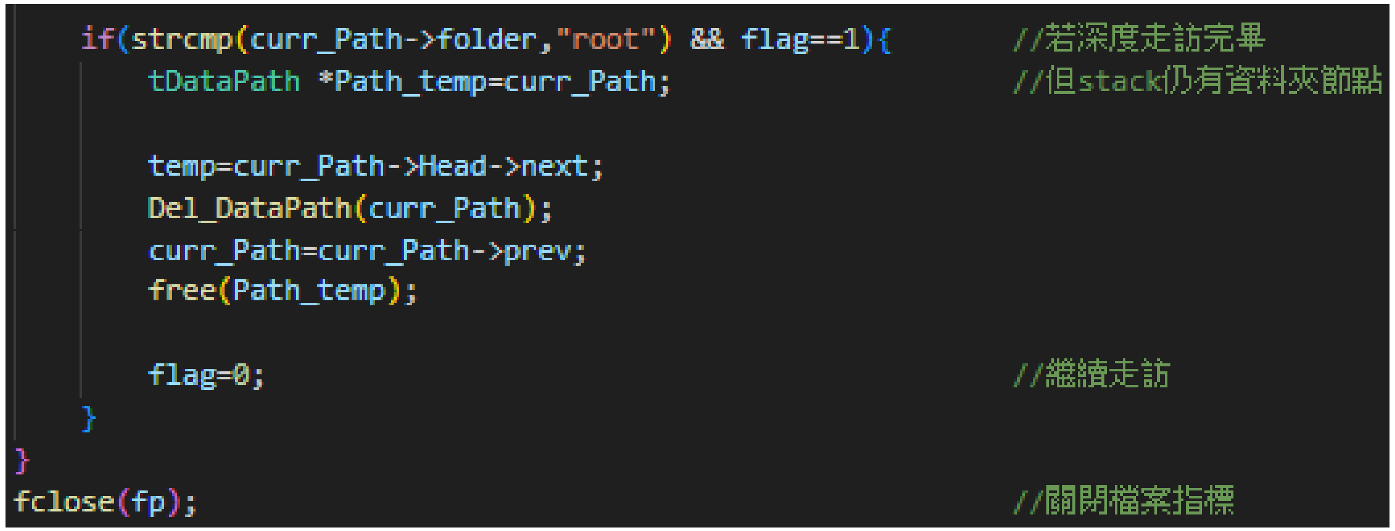
```c
if(prev->folder == 1){                           //若當前節點為資料夾
    strcpy(SaveTemp.Name,prev->FileName);
    SaveTemp.folder=1;
    SaveTemp.first=(prev->parent==NULL)?1:0;
    SaveTemp.finish=flag;

    if(prev->Left->next==NULL){                  //特殊值:為空
        SaveTemp.size=-1;
    }else{                                       //非空
        SaveTemp.size=0;
        Add_DataPath(curr_Path,prev->FileName,prev->Left);
        curr_Path=curr_Path->next;               //stack佇列
    }
    fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);  //檔案寫出
}else{                                           //若當前節點為檔案
    strcpy(SaveTemp.Name,prev->FileName);
    SaveTemp.folder=0;
    SaveTemp.first=(prev->parent==NULL)?1:0;
    SaveTemp.finish=flag;
    SaveTemp.size=prev->size;

    fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);  //結構寫出
    fwrite((void*)prev->content,prev->size,1,fp); //檔案寫出
    free(prev->content);                         //釋放content內容空間
}
free(prev);                                      //釋放資料節點空間
```

OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



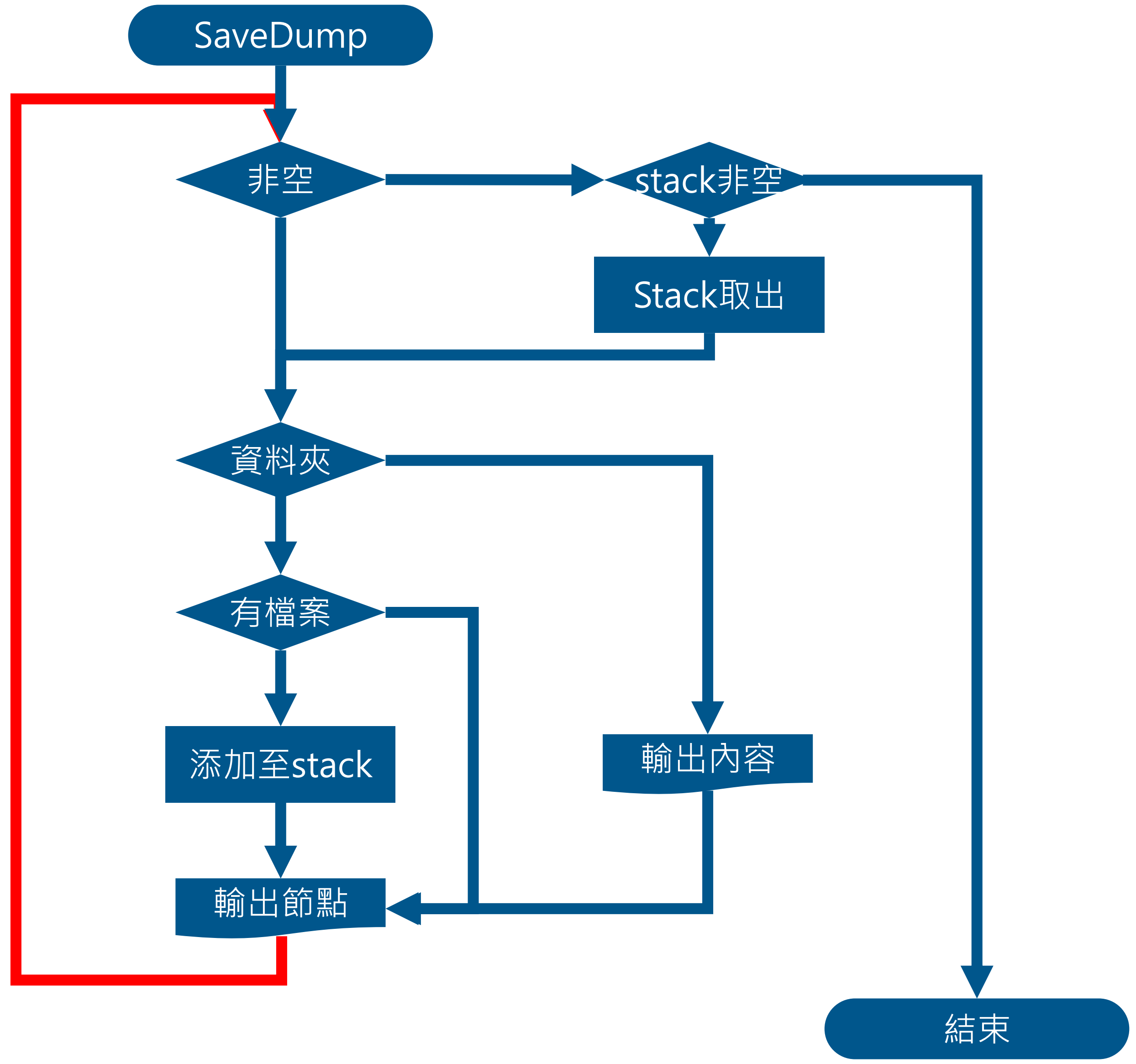
```
tDataPath *curr_Path=root;
tDataTree *temp=head->next, *prev;
int flag=0;
FILE *fp = fopen("my_fs.dump", "wb");

fwrite(&SizeOfPartition,sizeof(int),1,fp);        //儲存大小資訊
if(head->next==NULL){
    return;
}

while(flag == 0){
    tSaveFormat SaveTemp;       //Name[],folder?,first?,finish?,size?
    prev = temp;                                        //儲存當前節點

    if(temp->Right != NULL){                            //若非空
        temp = temp->Right;                              //繼續走訪
    }else{                                              //若空
        flag = 1;                                        //舉旗標，脫離while
    }

    if(prev->folder == 1){                              //若當前節點為資料夾
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=1;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;

        if(prev->Left->next==NULL){                     //特殊值:為空
            SaveTemp.size=-1;
        }else{                                          //非空
            SaveTemp.size=0;
            Add_DataPath(curr_Path,prev->FileName,prev->Left);
            curr_Path=curr_Path->next;                  //stack佇列
        }
        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);     //檔案寫出
    }else{                                              //若當前節點為檔案
        strcpy(SaveTemp.Name,prev->FileName);
        SaveTemp.folder=0;
        SaveTemp.first=(prev->parent==NULL)?1:0;
        SaveTemp.finish=flag;
        SaveTemp.size=prev->size;

        fwrite(&SaveTemp,sizeof(tSaveFormat),1,fp);     //結構寫出
        fwrite((void*)prev->content,prev->size,1,fp);   //檔案寫出
        free(prev->content);                            //釋放content內容空間
    }
    free(prev);                                         //釋放資料節點空間

    if(strcmp(curr_Path->folder,"root") && flag==1){    //若深度走訪完畢
        tDataPath *Path_temp=curr_Path;                 //但stack仍有資料夾節點

        temp=curr_Path->Head->next;
        Del_DataPath(curr_Path);
        curr_Path=curr_Path->prev;
        free(Path_temp);

        flag=0;                                         //繼續走訪
    }
}
fclose(fp);                                             //關閉檔案指標
```



```
    if(strcmp(curr_Path->folder,"root") && flag==1){        //若深度走訪完畢
        tDataPath *Path_temp=curr_Path;                     //但stack仍有資料夾節點

        temp=curr_Path->Head->next;
        Del_DataPath(curr_Path);
        curr_Path=curr_Path->prev;
        free(Path_temp);

        flag=0;                                             //繼續走訪
    }
}
fclose(fp);                                                 //關閉檔案指標
```

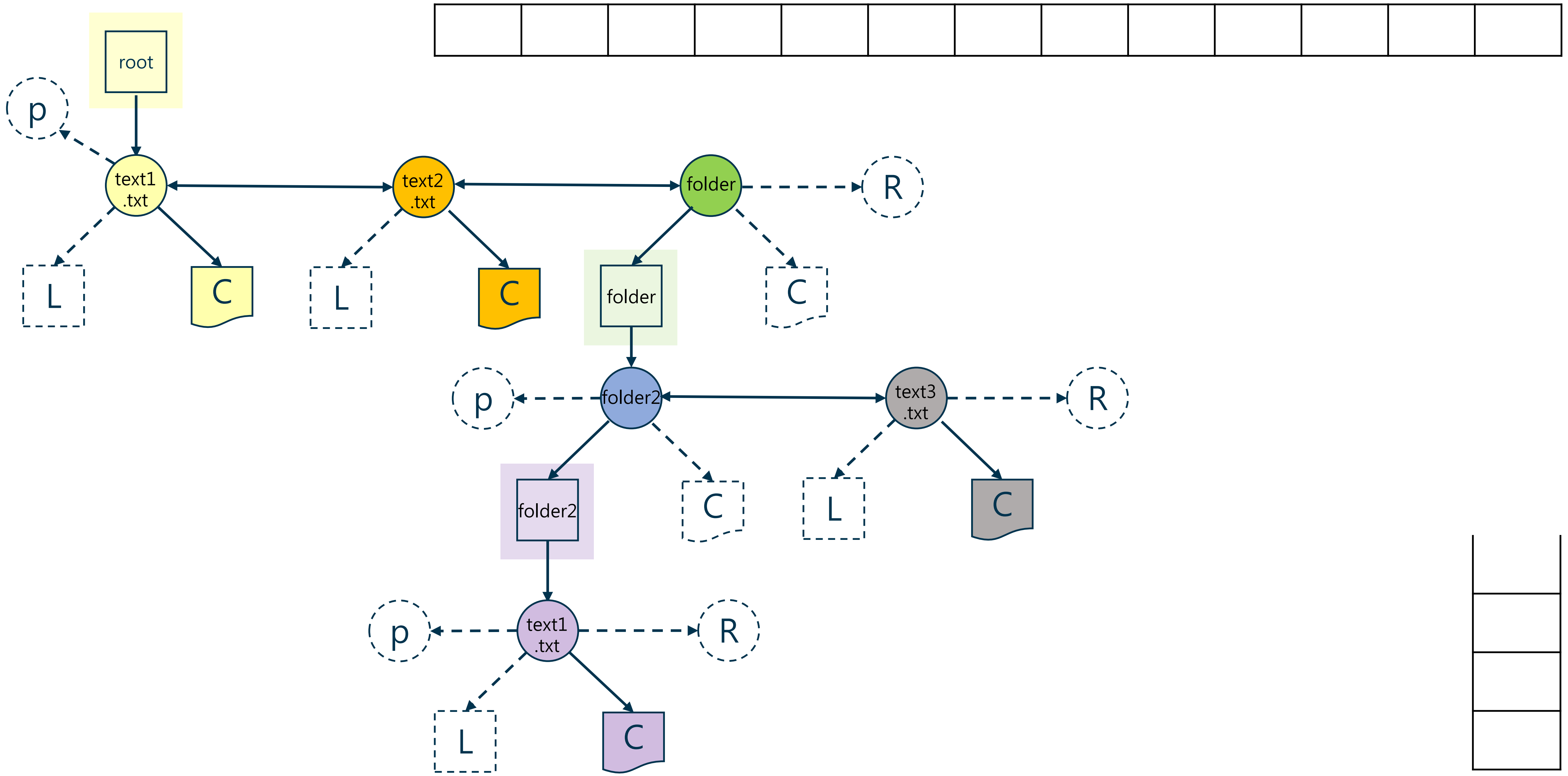
OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



OPER_SaveDump

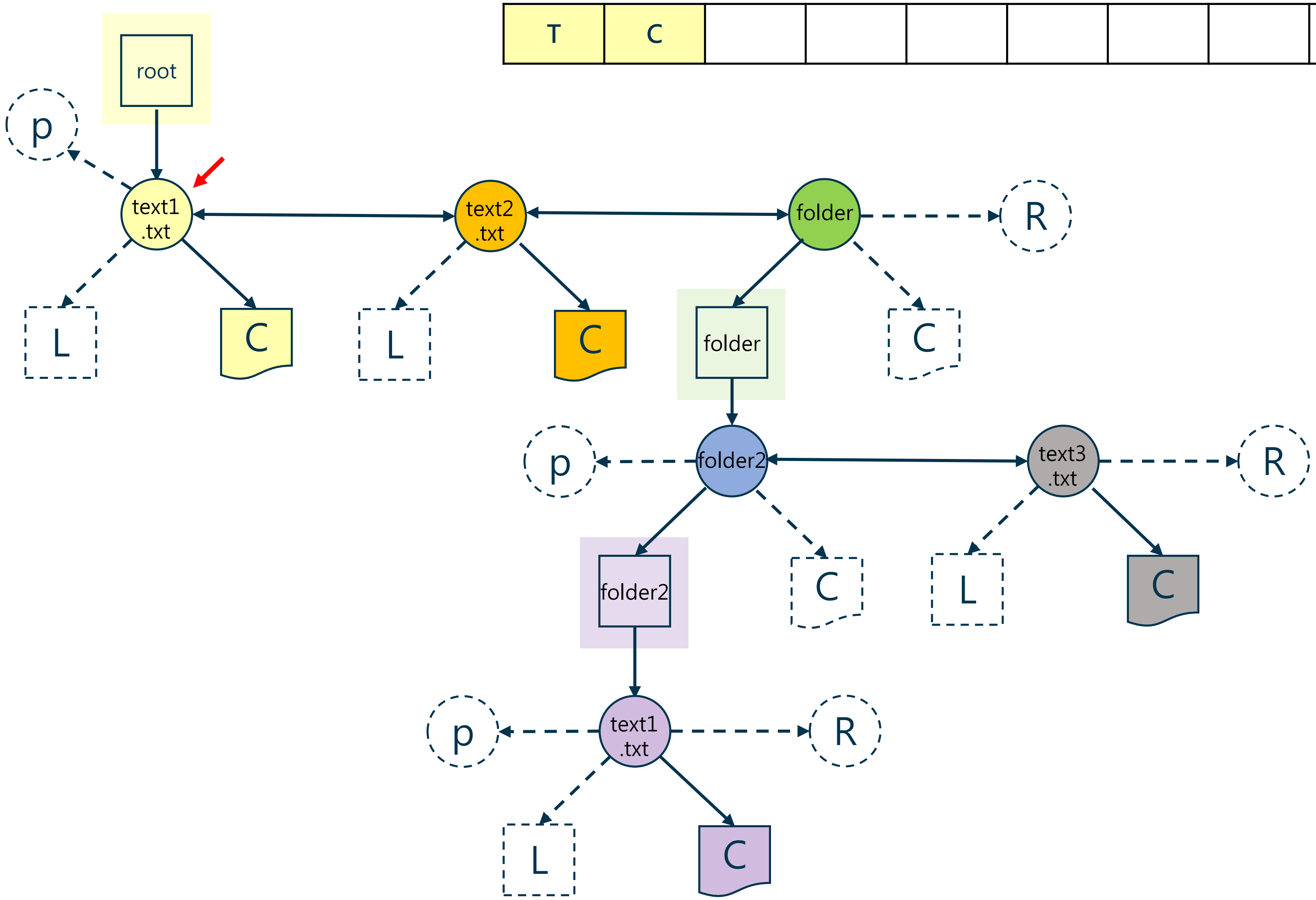
```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```

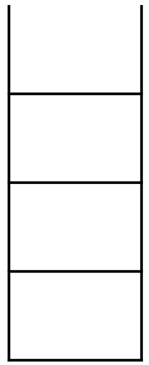


OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```

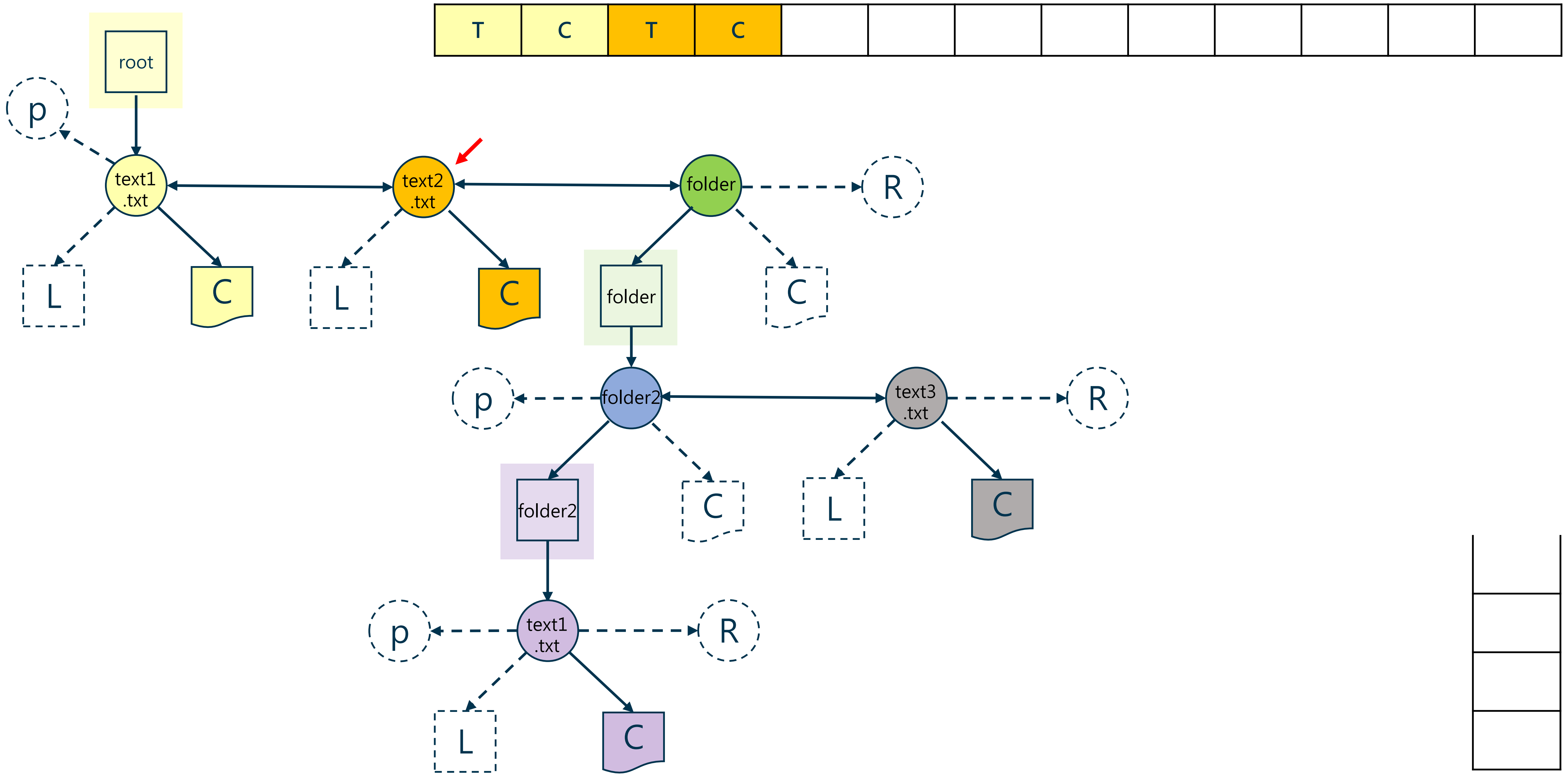
T	c											





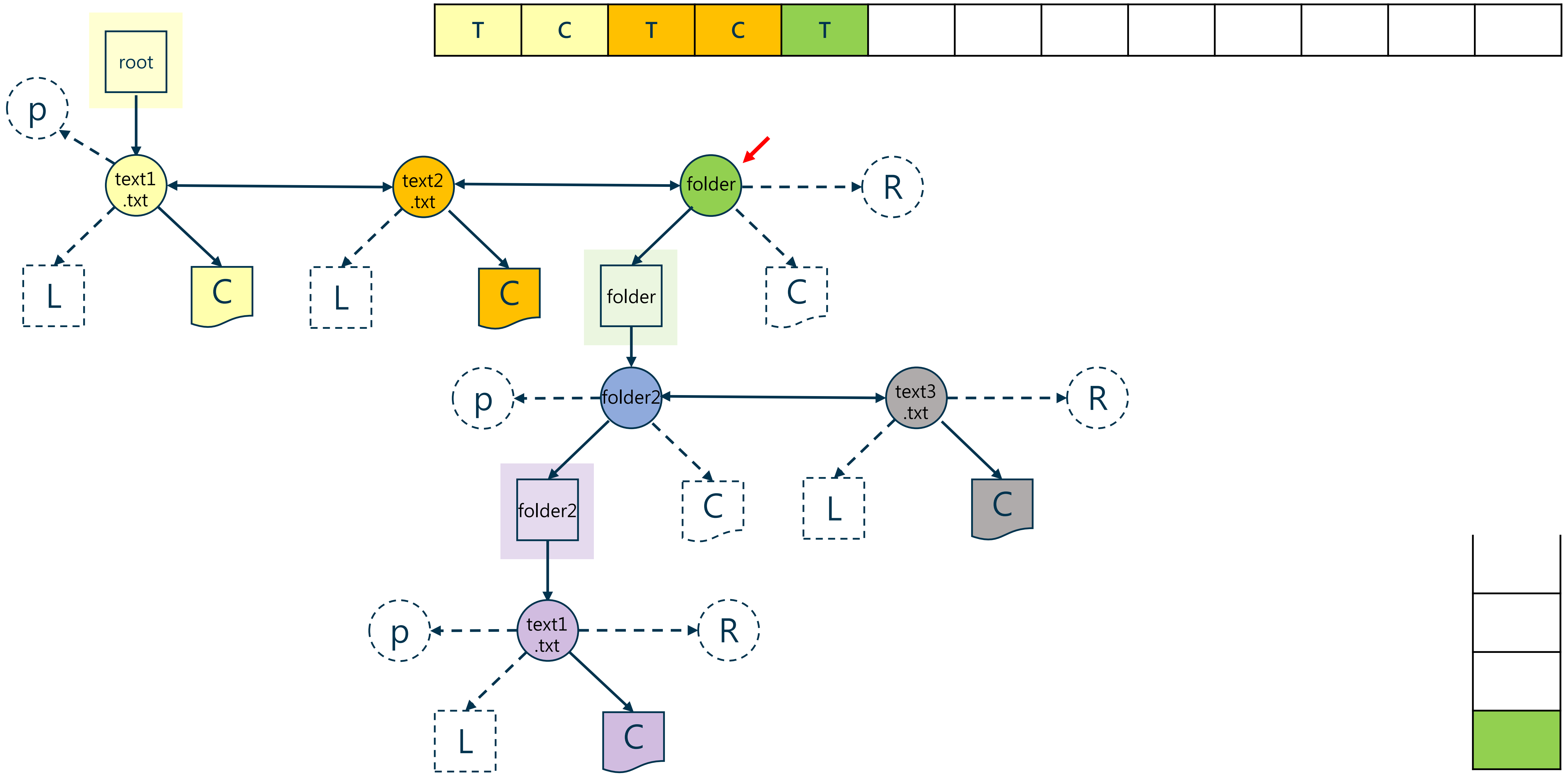
OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



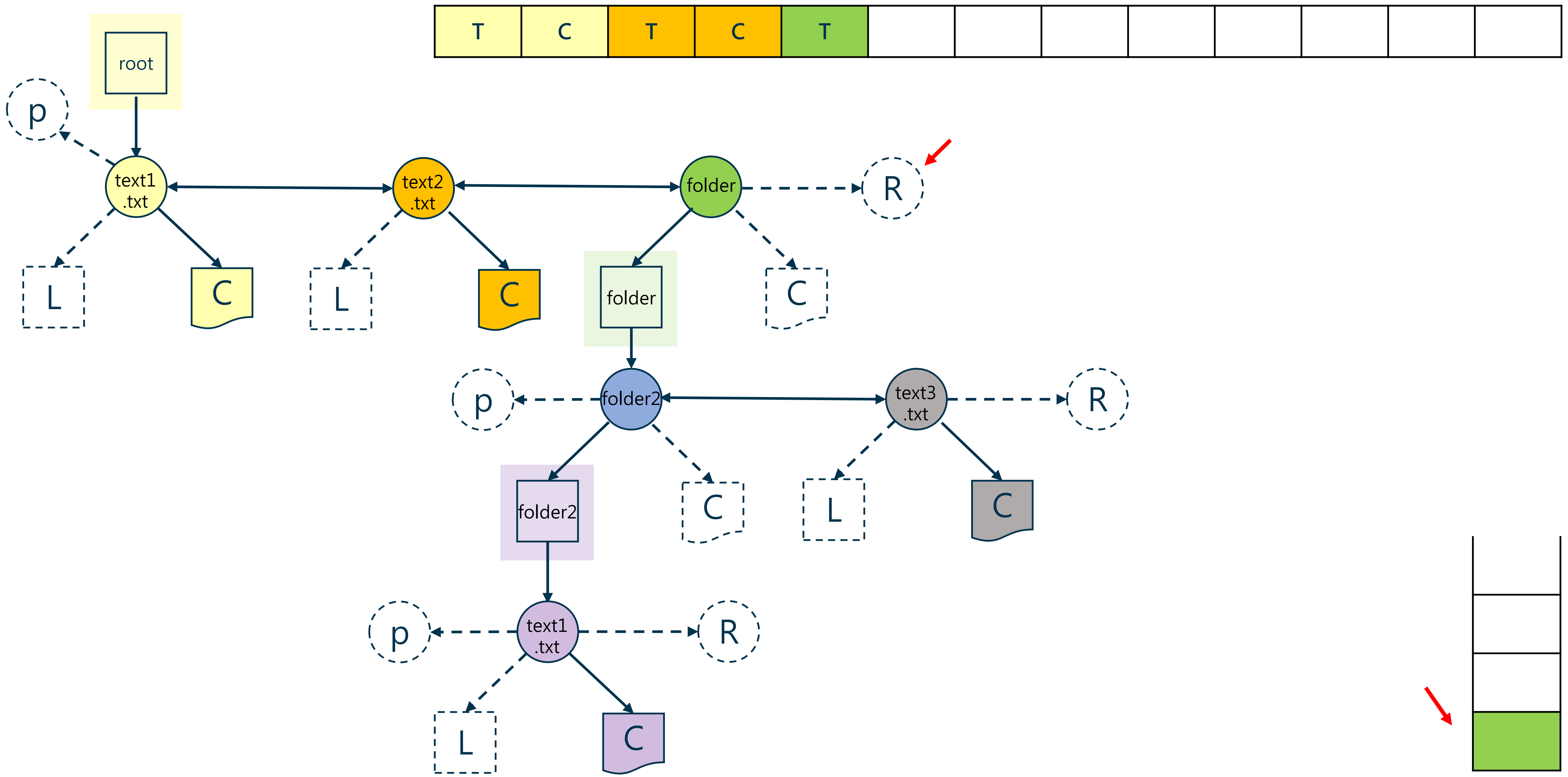
OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



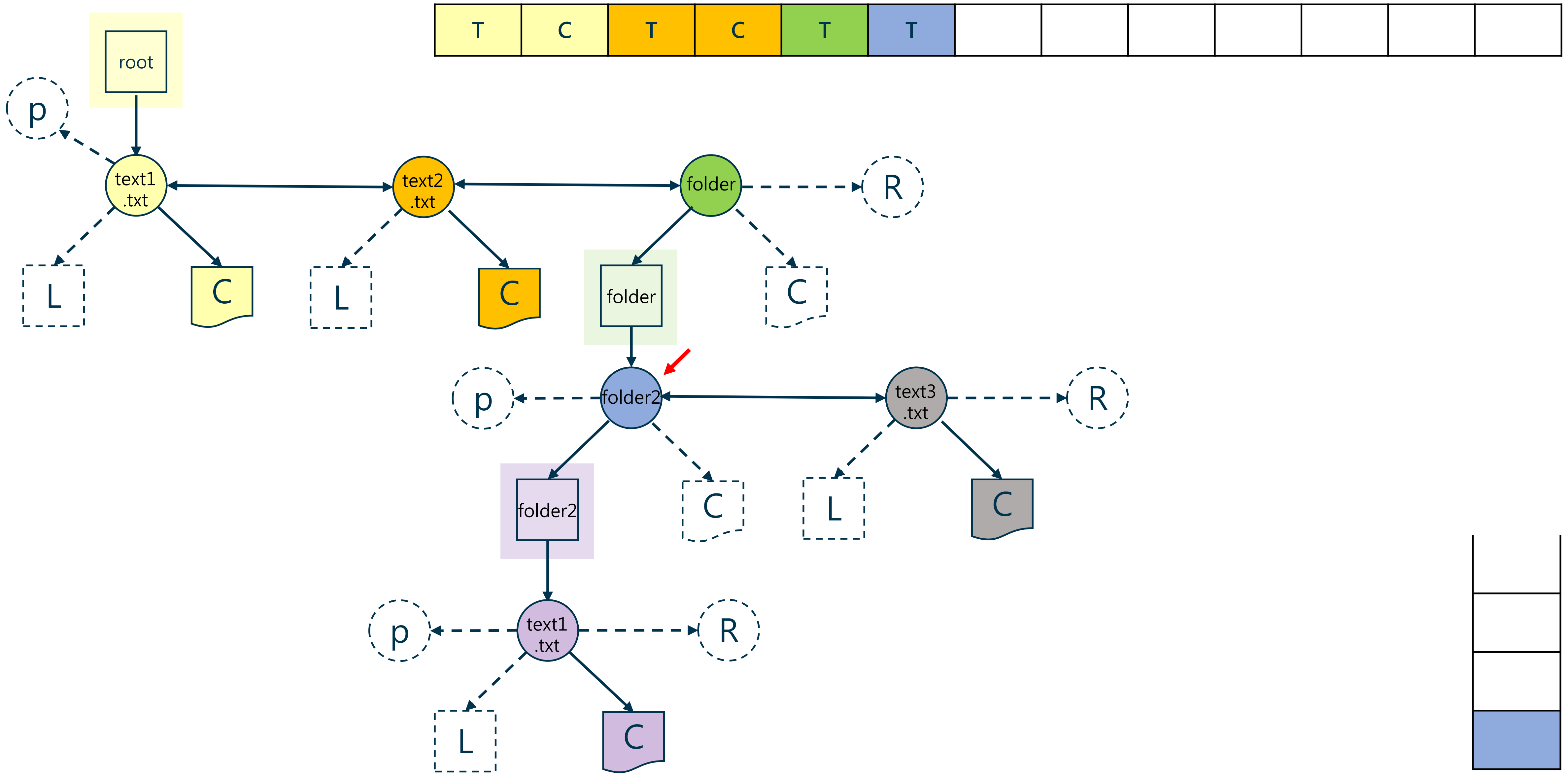
OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



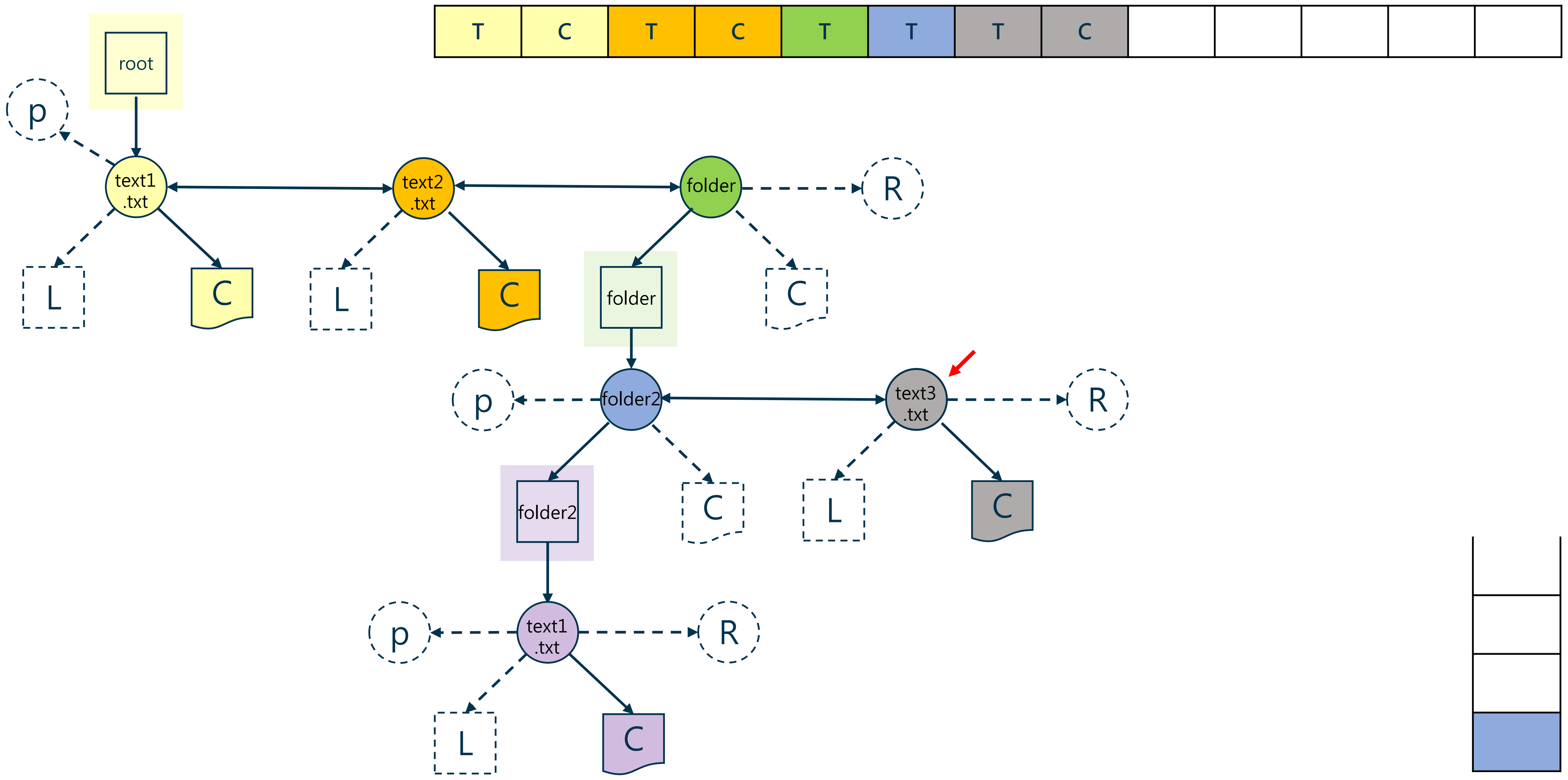
OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



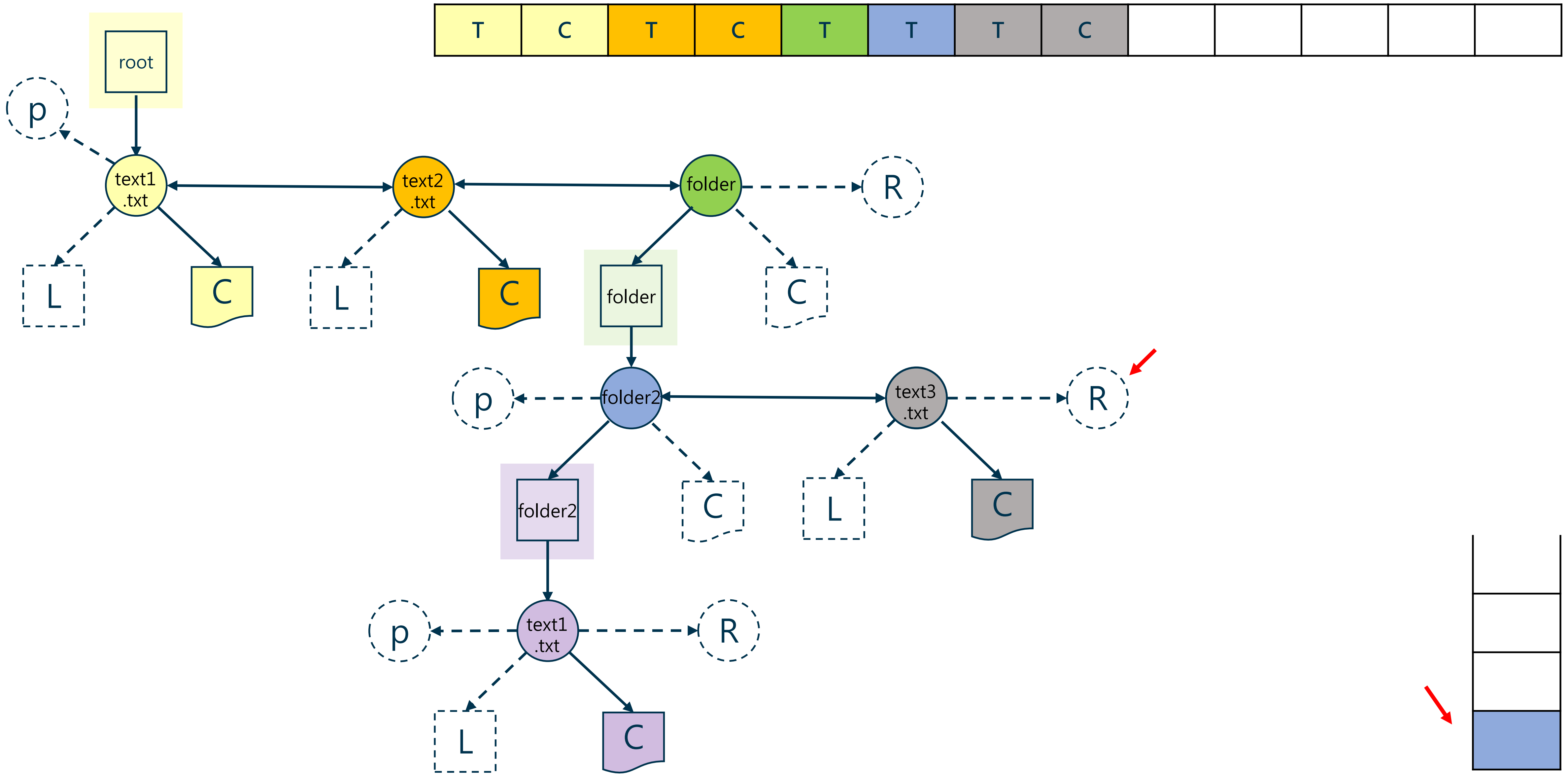
OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```



OPER_SaveDump

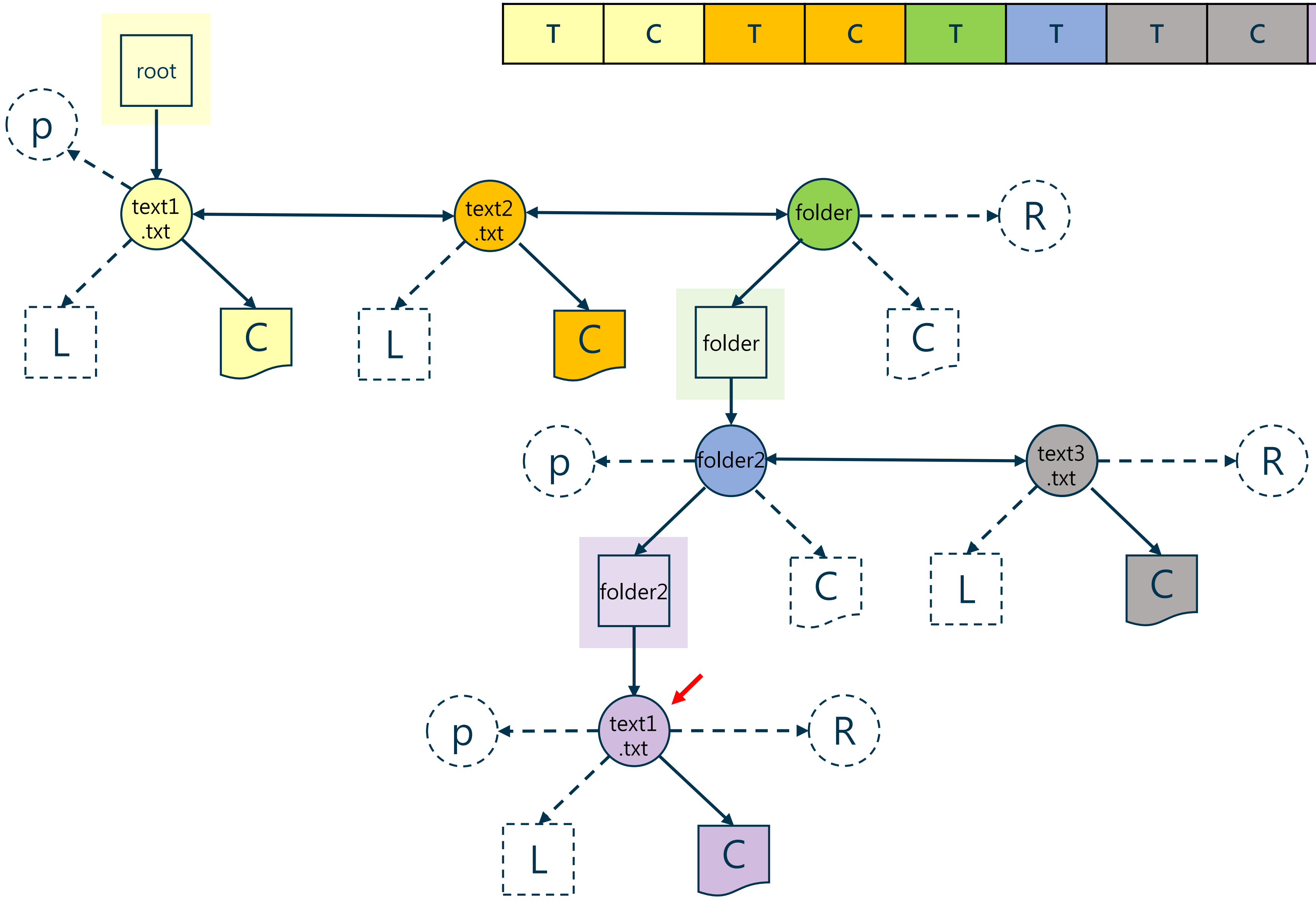
```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```

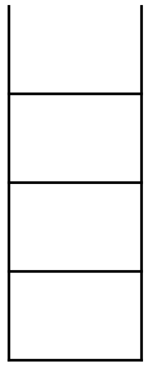


OPER_SaveDump

```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```

T	C	T	C	T	T	T	C	T	C			

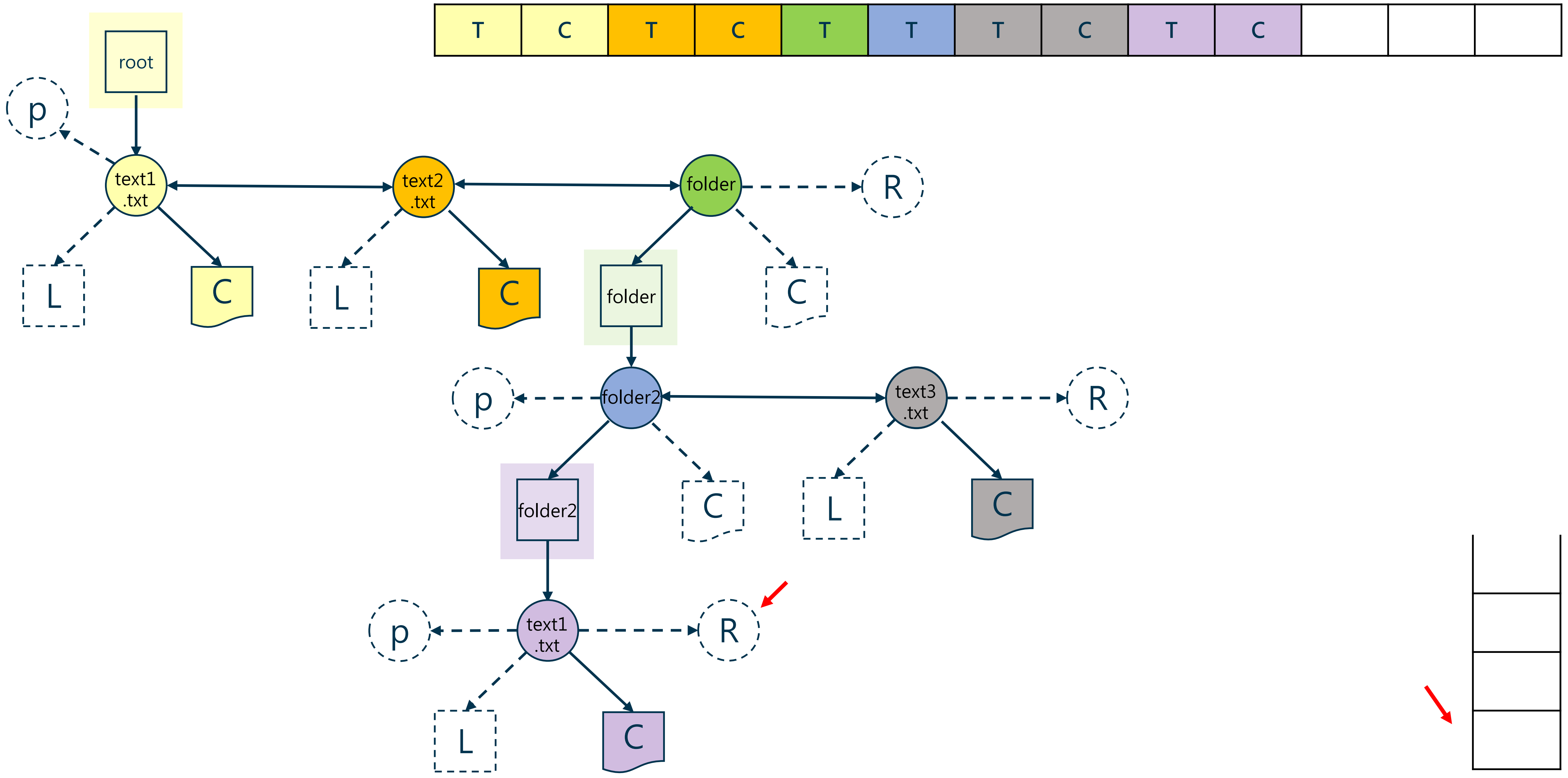




OPER_SaveDump

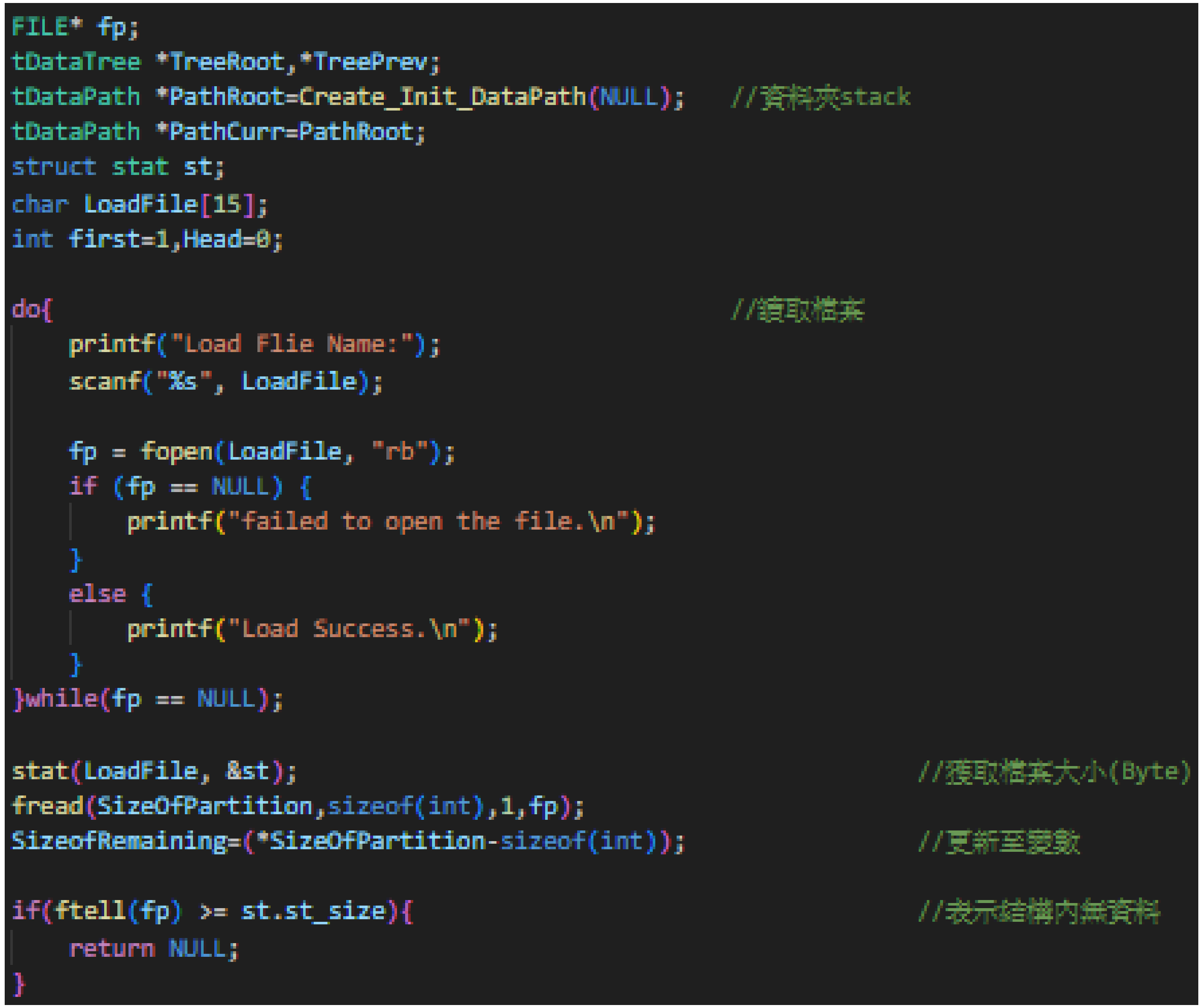
```
void OPER_SaveDump(tDataHead *head,int SizeOfPartition,tDataPath *root);
OPER_SaveDump(head,SizeOfPartition,root);
```

T	C	T	C	T	T	T	C	T	C			



OPER_LoadDump

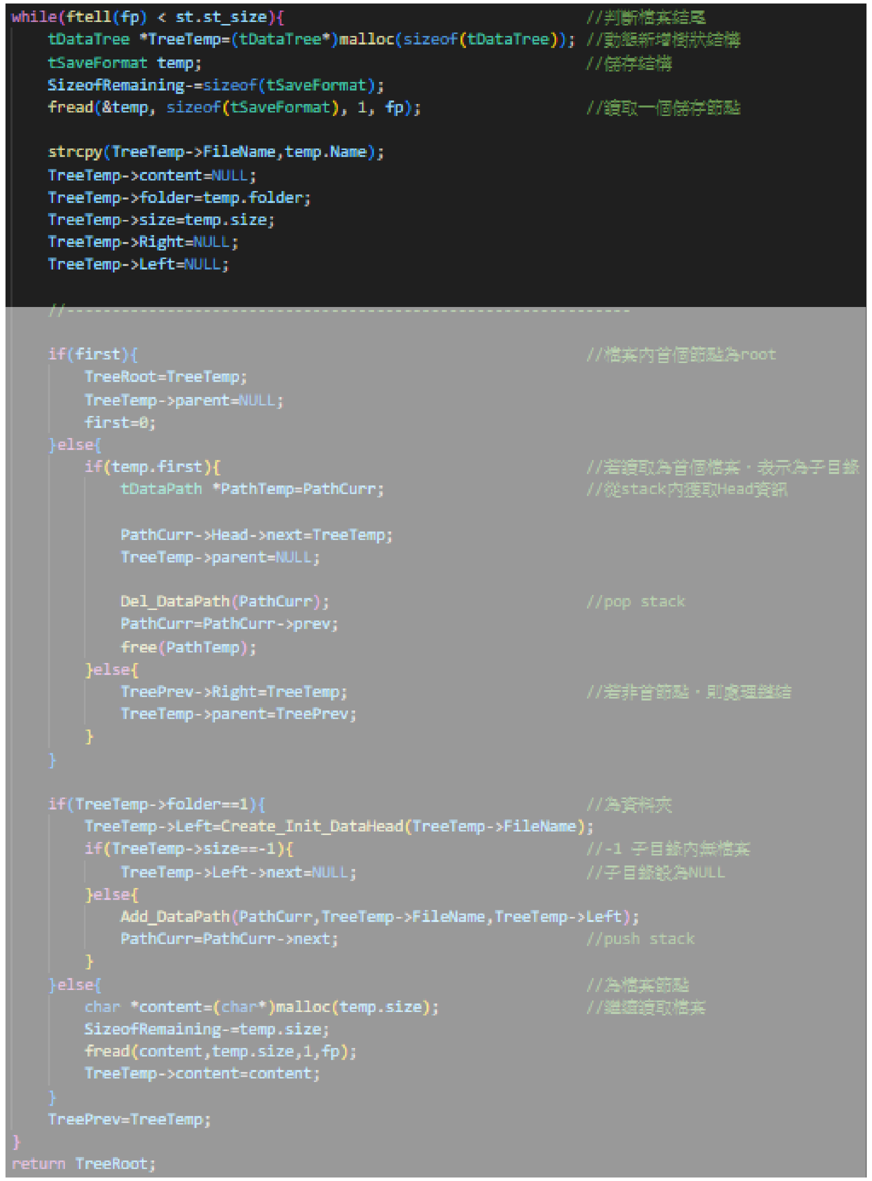
```
tDataTree* OPER_LoadDump(int *SizeOfPartition);
tDataTree *load=OPER_LoadDump(SizeOfPartition);
```



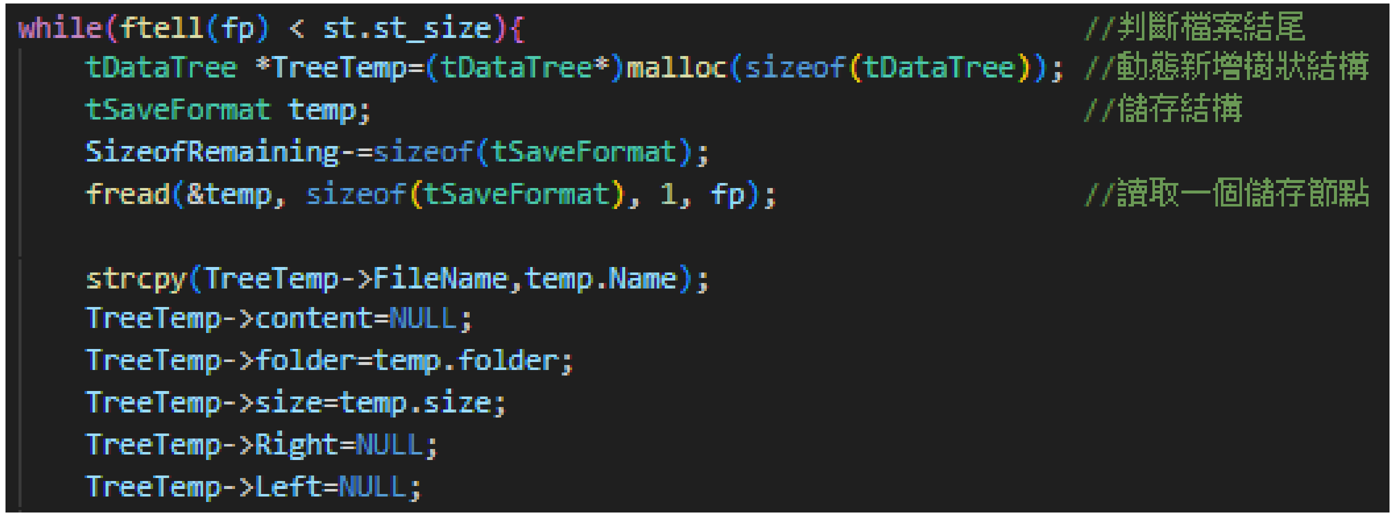
```
FILE* fp;
tDataTree *TreeRoot,*TreePrev;
tDataPath *PathRoot=Create_Init_DataPath(NULL);    //資料夾stack
tDataPath *PathCurr=PathRoot;
struct stat st;
char LoadFile[15];
int first=1,Head=0;

do{                                                //讀取檔案
    printf("Load Flie Name:");
    scanf("%s", LoadFile);

    fp = fopen(LoadFile, "rb");
    if (fp == NULL) {
        printf("failed to open the file.\n");
    }
    else {
        printf("Load Success.\n");
    }
}while(fp == NULL);

stat(LoadFile, &st);                                            //獲取檔案大小(Byte)
fread(SizeOfPartition,sizeof(int),1,fp);
SizeofRemaining=(*SizeOfPartition-sizeof(int));                 //更新至變數

if(ftell(fp) >= st.st_size){                                    //表示結構內無資料
    return NULL;
}
```

OPER_LoadDump

```c
tDataTree* OPER_LoadDump(int *SizeOfPartition);
tDataTree *load=OPER_LoadDump(SizeOfPartition);
```



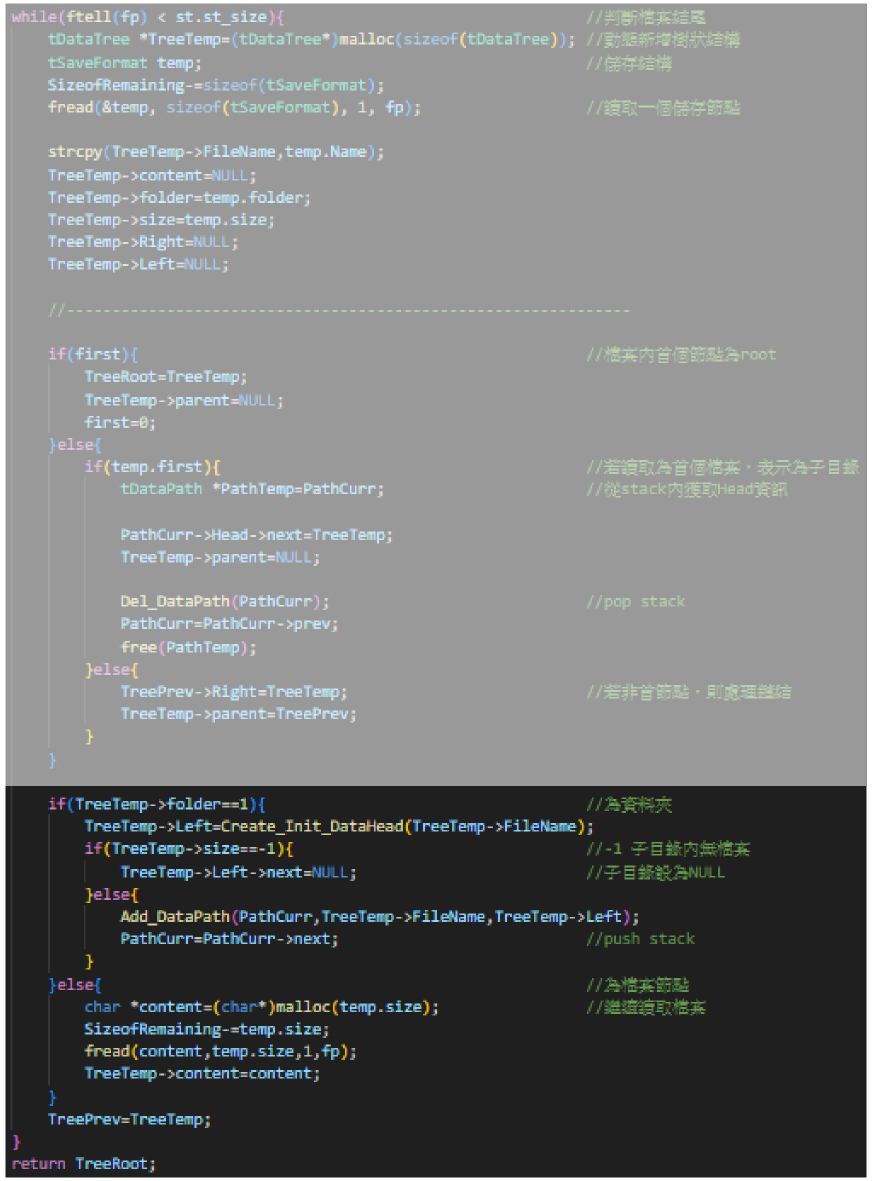
```c
while(ftell(fp) < st.st_size){                                       //判斷檔案結尾
    tDataTree *TreeTemp=(tDataTree*)malloc(sizeof(tDataTree)); //動態新增樹狀結構
    tSaveFormat temp;                                                //儲存結構
    SizeofRemaining-=sizeof(tSaveFormat);
    fread(&temp, sizeof(tSaveFormat), 1, fp);                        //讀取一個儲存節點

    strcpy(TreeTemp->FileName,temp.Name);
    TreeTemp->content=NULL;
    TreeTemp->folder=temp.folder;
    TreeTemp->size=temp.size;
    TreeTemp->Right=NULL;
    TreeTemp->Left=NULL;

    //------------------------------------------------------------------

    if(first){                                                       //檔案內首個節點為root
        TreeRoot=TreeTemp;
        TreeTemp->parent=NULL;
        first=0;
    }else{
        if(temp.first){                                              //若讀取為首個檔案，表示為子目錄
            tDataPath *PathTemp=PathCurr;                            //從stack內獲取Head資訊

            PathCurr->Head->next=TreeTemp;
            TreeTemp->parent=NULL;

            Del_DataPath(PathCurr);                                  //pop stack
            PathCurr=PathCurr->prev;
            free(PathTemp);
        }else{
            TreePrev->Right=TreeTemp;                                //若非首節點，則處理鏈結
            TreeTemp->parent=TreePrev;
        }
    }

    if(TreeTemp->folder==1){                                         //為資料夾
        TreeTemp->Left=Create_Init_DataHead(TreeTemp->FileName);
        if(TreeTemp->size==-1){                                      //-1 子目錄內無檔案
            TreeTemp->Left->next=NULL;                               //子目錄設為NULL
        }else{
            Add_DataPath(PathCurr,TreeTemp->FileName,TreeTemp->Left);
            PathCurr=PathCurr->next;                                 //push stack
        }
    }else{                                                           //為檔案節點
        char *content=(char*)malloc(temp.size);                      //繼續讀取檔案
        SizeofRemaining-=temp.size;
        fread(content,temp.size,1,fp);
        TreeTemp->content=content;
    }
    TreePrev=TreeTemp;
}
return TreeRoot;
```



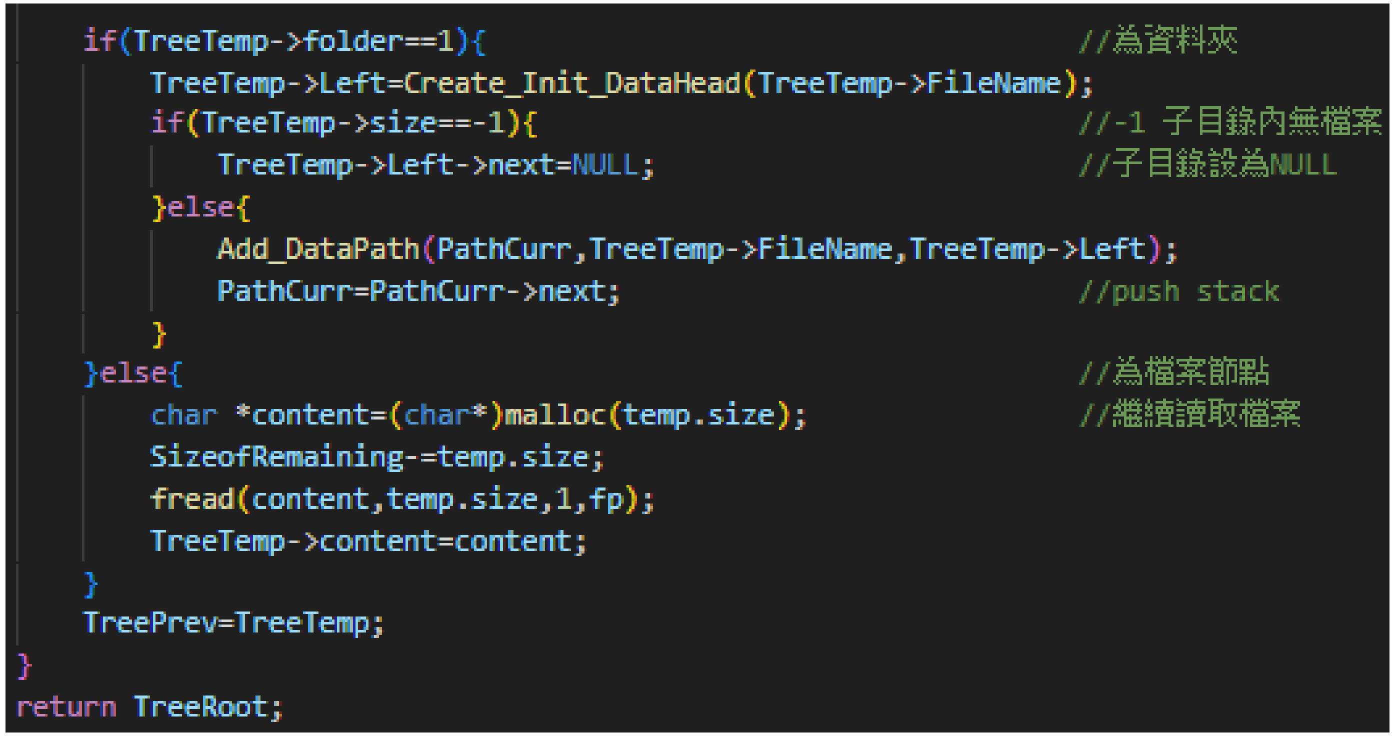
```c
while(ftell(fp) < st.st_size){                                       //判斷檔案結尾
    tDataTree *TreeTemp=(tDataTree*)malloc(sizeof(tDataTree)); //動態新增樹狀結構
    tSaveFormat temp;                                                //儲存結構
    SizeofRemaining-=sizeof(tSaveFormat);
    fread(&temp, sizeof(tSaveFormat), 1, fp);                        //讀取一個儲存節點

    strcpy(TreeTemp->FileName,temp.Name);
    TreeTemp->content=NULL;
    TreeTemp->folder=temp.folder;
    TreeTemp->size=temp.size;
    TreeTemp->Right=NULL;
    TreeTemp->Left=NULL;
```

OPER_LoadDump

```c
tDataTree* OPER_LoadDump(int *SizeOfPartition);
tDataTree *load=OPER_LoadDump(SizeOfPartition);
```

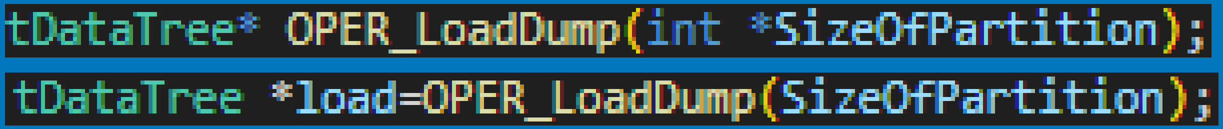


```c
while(ftell(fp) < st.st_size){                          //判斷檔案結尾
    tDataTree *TreeTemp=(tDataTree*)malloc(sizeof(tDataTree)); //動態新增樹狀結構
    tSaveFormat temp;                                   //儲存結構
    SizeofRemaining-=sizeof(tSaveFormat);
    fread(&temp, sizeof(tSaveFormat), 1, fp);           //讀取一個儲存節點

    strcpy(TreeTemp->FileName,temp.Name);
    TreeTemp->content=NULL;
    TreeTemp->folder=temp.folder;
    TreeTemp->size=temp.size;
    TreeTemp->Right=NULL;
    TreeTemp->Left=NULL;

    //----------------------------------------------------------------

    if(first){                                          //檔案內首個節點為root
        TreeRoot=TreeTemp;
        TreeTemp->parent=NULL;
        first=0;
    }else{
        if(temp.first){                                 //若讀取為首個檔案，表示為子目錄
            tDataPath *PathTemp=PathCurr;               //從stack內獲取Head資訊

            PathCurr->Head->next=TreeTemp;
            TreeTemp->parent=NULL;

            Del_DataPath(PathCurr);                     //pop stack
            PathCurr=PathCurr->prev;
            free(PathTemp);
        }else{
            TreePrev->Right=TreeTemp;                   //若非首節點，則處理鏈結
            TreeTemp->parent=TreePrev;
        }
    }
    if(TreeTemp->folder==1){                            //為資料夾
        TreeTemp->Left=Create_Init_DataHead(TreeTemp->FileName);
        if(TreeTemp->size==-1){                         //-1 子目錄內無檔案
            TreeTemp->Left->next=NULL;                  //子目錄設為NULL
        }else{
            Add_DataPath(PathCurr,TreeTemp->FileName,TreeTemp->Left);
            PathCurr=PathCurr->next;                    //push stack
        }
    }else{                                              //為檔案節點
        char *content=(char*)malloc(temp.size);         //繼續讀取檔案
        SizeofRemaining-=temp.size;
        fread(content,temp.size,1,fp);
        TreeTemp->content=content;
    }
    TreePrev=TreeTemp;
}
return TreeRoot;
```

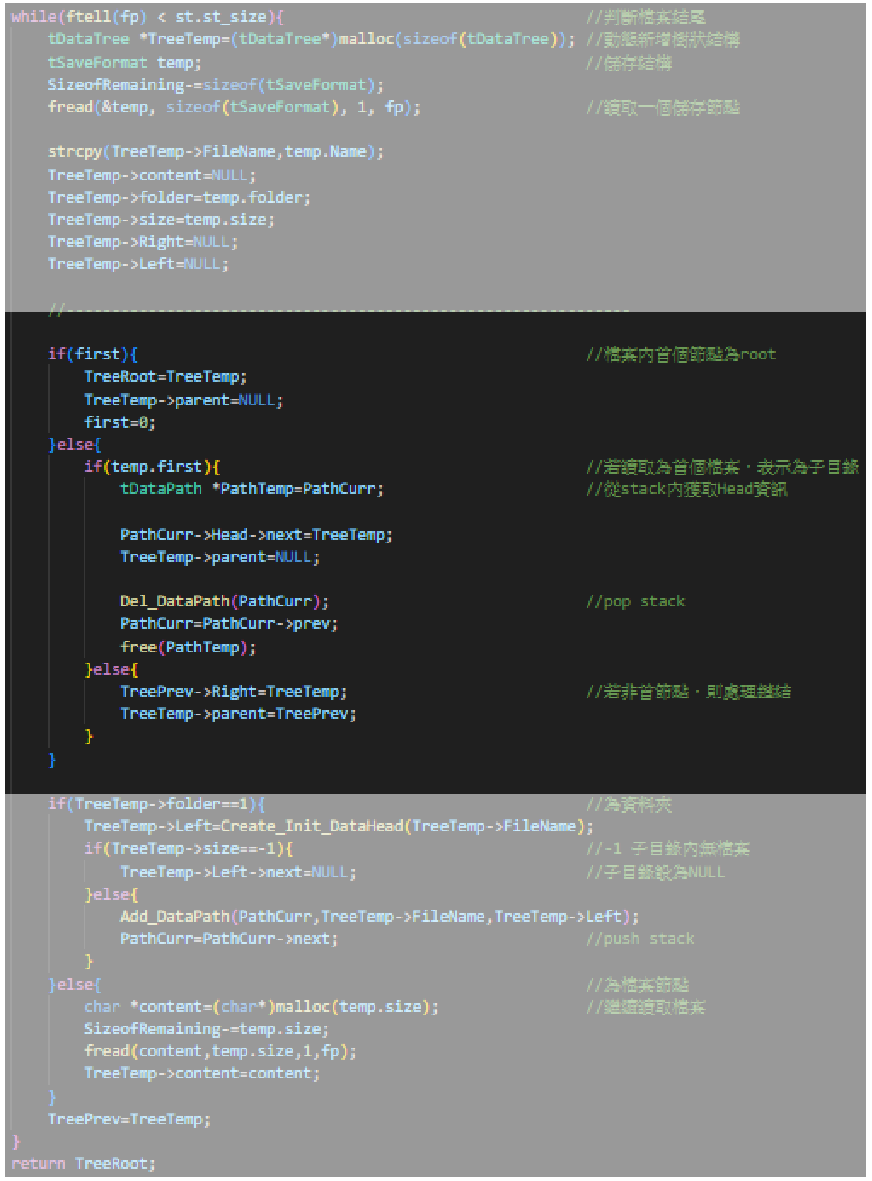


```c
    if(TreeTemp->folder==1){                            //為資料夾
        TreeTemp->Left=Create_Init_DataHead(TreeTemp->FileName);
        if(TreeTemp->size==-1){                         //-1 子目錄內無檔案
            TreeTemp->Left->next=NULL;                  //子目錄設為NULL
        }else{
            Add_DataPath(PathCurr,TreeTemp->FileName,TreeTemp->Left);
            PathCurr=PathCurr->next;                    //push stack
        }
    }else{                                              //為檔案節點
        char *content=(char*)malloc(temp.size);         //繼續讀取檔案
        SizeofRemaining-=temp.size;
        fread(content,temp.size,1,fp);
        TreeTemp->content=content;
    }
    TreePrev=TreeTemp;
}
return TreeRoot;
```

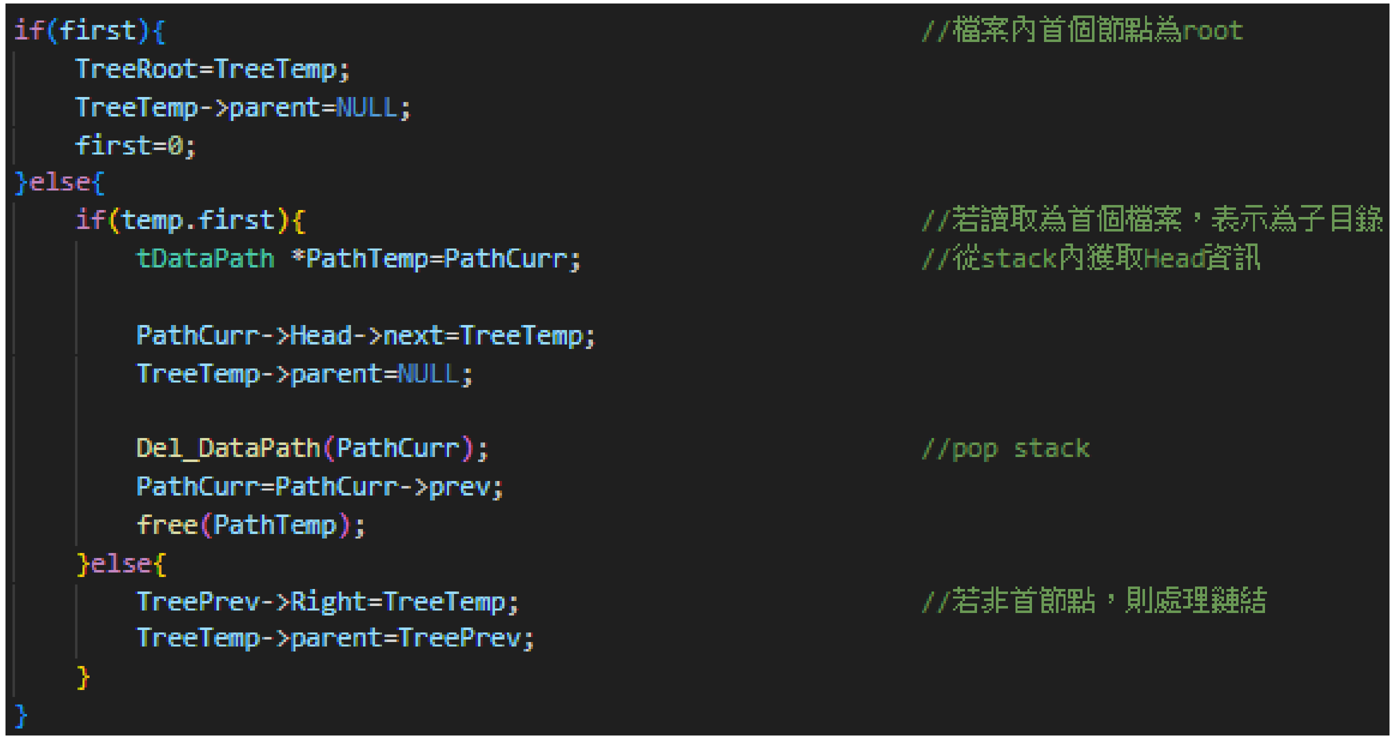
OPER_LoadDump



```c
tDataTree* OPER_LoadDump(int *SizeOfPartition);
tDataTree *load=OPER_LoadDump(SizeOfPartition);
```

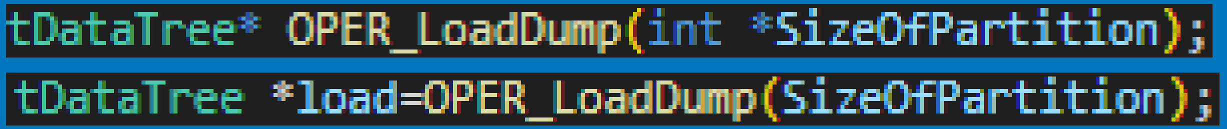


```c
while(ftell(fp) < st.st_size){                                      //判斷檔案結尾
    tDataTree *TreeTemp=(tDataTree*)malloc(sizeof(tDataTree));  //動態新增樹狀結構
    tSaveFormat temp;                                           //儲存結構
    SizeofRemaining-=sizeof(tSaveFormat);
    fread(&temp, sizeof(tSaveFormat), 1, fp);                   //讀取一個儲存節點

    strcpy(TreeTemp->FileName,temp.Name);
    TreeTemp->content=NULL;
    TreeTemp->folder=temp.folder;
    TreeTemp->size=temp.size;
    TreeTemp->Right=NULL;
    TreeTemp->Left=NULL;

    //-----------------------------------------------------------------

    if(first){                                                  //檔案內首個節點為root
        TreeRoot=TreeTemp;
        TreeTemp->parent=NULL;
        first=0;
    }else{
        if(temp.first){                                         //若讀取為首個檔案，表示為子目錄
            tDataPath *PathTemp=PathCurr;                       //從stack內獲取Head資訊

            PathCurr->Head->next=TreeTemp;
            TreeTemp->parent=NULL;

            Del_DataPath(PathCurr);                             //pop stack
            PathCurr=PathCurr->prev;
            free(PathTemp);
        }else{
            TreePrev->Right=TreeTemp;                           //若非首節點，則處理鏈結
            TreeTemp->parent=TreePrev;
        }
    }

    if(TreeTemp->folder==1){                                    //為資料夾
        TreeTemp->Left=Create_Init_DataHead(TreeTemp->FileName);
        if(TreeTemp->size==-1){                                 //-1 子目錄內無檔案
            TreeTemp->Left->next=NULL;                          //子目錄設為NULL
        }else{
            Add_DataPath(PathCurr,TreeTemp->FileName,TreeTemp->Left);
            PathCurr=PathCurr->next;                            //push stack
        }
    }else{                                                      //為檔案節點
        char *content=(char*)malloc(temp.size);                 //繼續讀取檔案
        SizeofRemaining-=temp.size;
        fread(content,temp.size,1,fp);
        TreeTemp->content=content;
    }
    TreePrev=TreeTemp;
}
return TreeRoot;
```

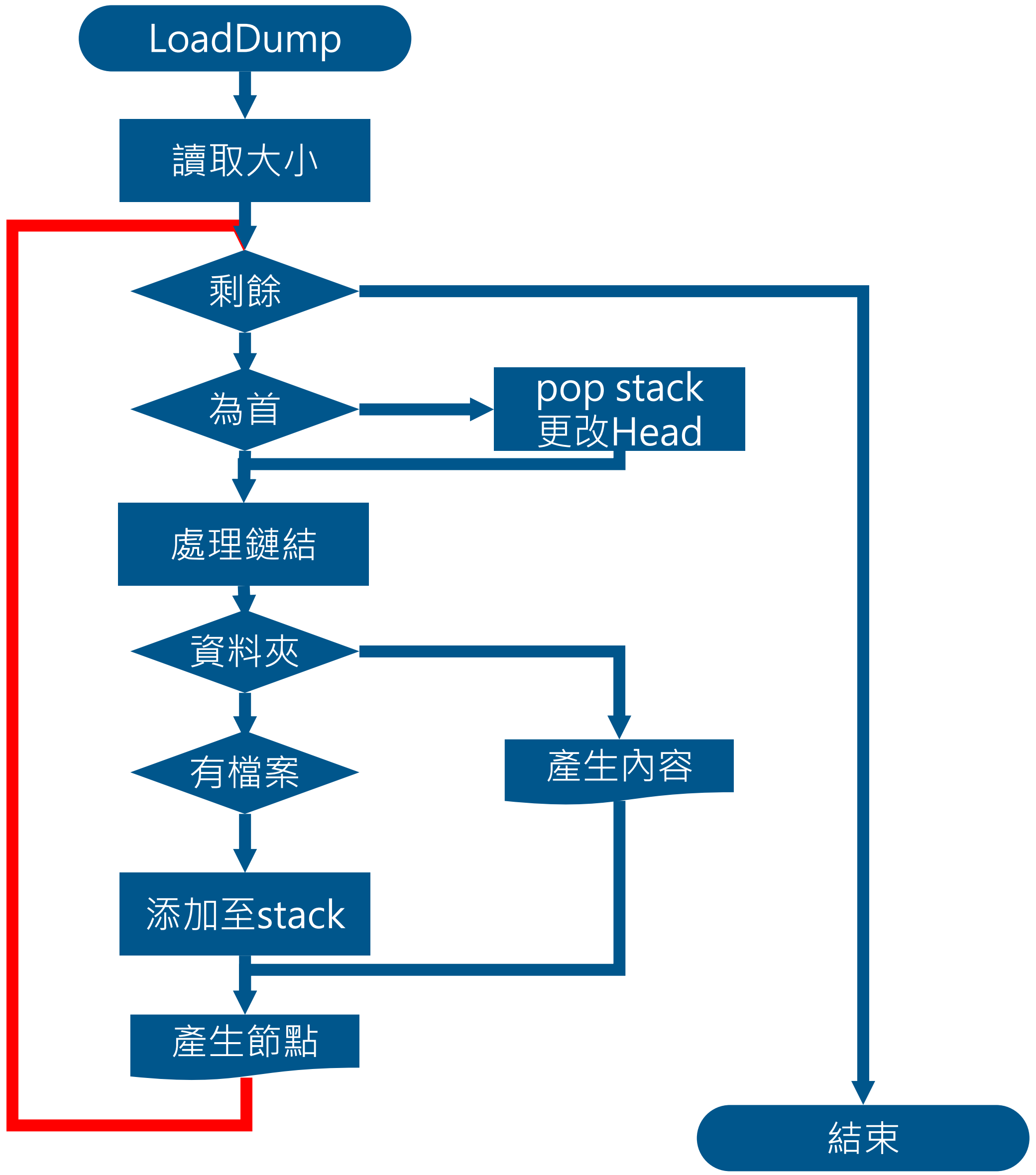


```c
if(first){                                                  //檔案內首個節點為root
    TreeRoot=TreeTemp;
    TreeTemp->parent=NULL;
    first=0;
}else{
    if(temp.first){                                         //若讀取為首個檔案，表示為子目錄
        tDataPath *PathTemp=PathCurr;                       //從stack內獲取Head資訊

        PathCurr->Head->next=TreeTemp;
        TreeTemp->parent=NULL;

        Del_DataPath(PathCurr);                             //pop stack
        PathCurr=PathCurr->prev;
        free(PathTemp);
    }else{
        TreePrev->Right=TreeTemp;                           //若非首節點，則處理鏈結
        TreeTemp->parent=TreePrev;
    }
}
```

OPER_LoadDump



```
tDataTree* OPER_LoadDump(int *SizeOfPartition);
tDataTree *load=OPER_LoadDump(SizeOfPartition);
```



THANK YOU