ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

10η Εβδομάδα: Διπλά Διασυνδεδεμένες Λίστες και Δείκτες σε Συνάρτηση

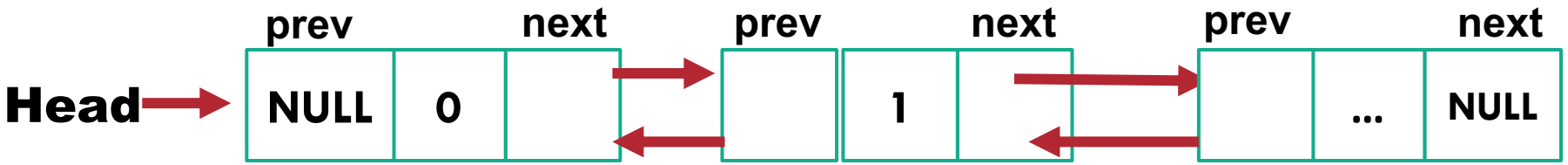
Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Διπλά διασυνδεδεμένη λίστα

```
typedef struct node {
                int AM;
                struct node *prev;
                struct node *next;
} STUDENT;
```

Έχοντας λοιπόν καθορίσει τη μορφή ενός κόμβου μπορούμε να φανταστούμε πως θα είναι μία διπλά συνδεδεμένη λίστα με κόμβους τύπου struct node που ορίσαμε νωρίτερα:



Τι κάνει το παρακάτω πρόγραμμα;

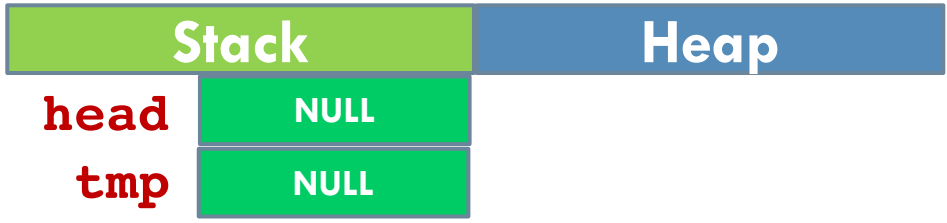
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
    int AM;
    struct node *prev;
    struct node *next;
}  STUDENT;


STUDENT * createDoubleLinkedList(int );
STUDENT* create_student_node(int AM);
void printlist(STUDENT *nd);

int main()
{
     STUDENT *head=NULL;
     head=createDoubleLinkedList(10);
     printlist(head);
     return 0;

}
```

Τι κάνει το παρακάτω πρόγραμμα;

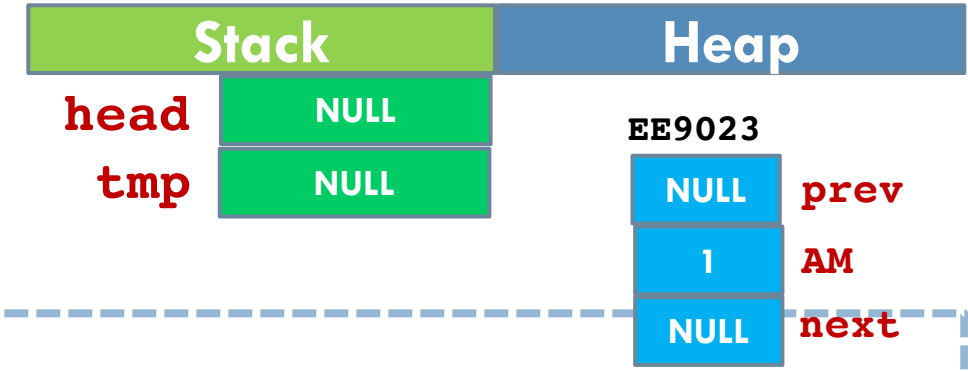
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node
{
    int AM;
    struct node *prev;
    struct node *next;
}  STUDENT;


STUDENT * createDoubleLinkedList(int );
STUDENT* create_student_node(int AM);
void printlist(STUDENT *nd);

int main()
{
     STUDENT *head=NULL;
     head=createDoubleLinkedList(10);
     printlist(head);
     return 0;

}
```

Δημιουργία Κόμβου Φοιτητής

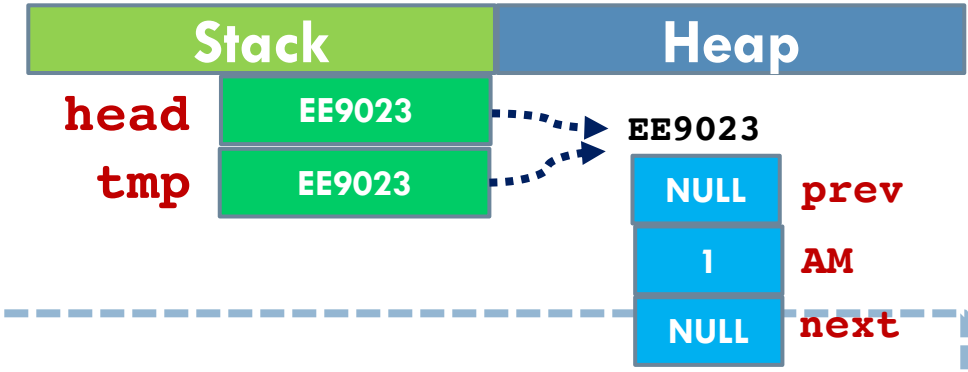

```
STUDENT* create_student_node(int AM) {
        STUDENT *node;
        node = (STUDENT *)malloc(sizeof(STUDENT));
        if (node == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else{
          node->prev = NULL;
          node->AM = AM;
          node->next = NULL;
          return node;
        }
    }
```



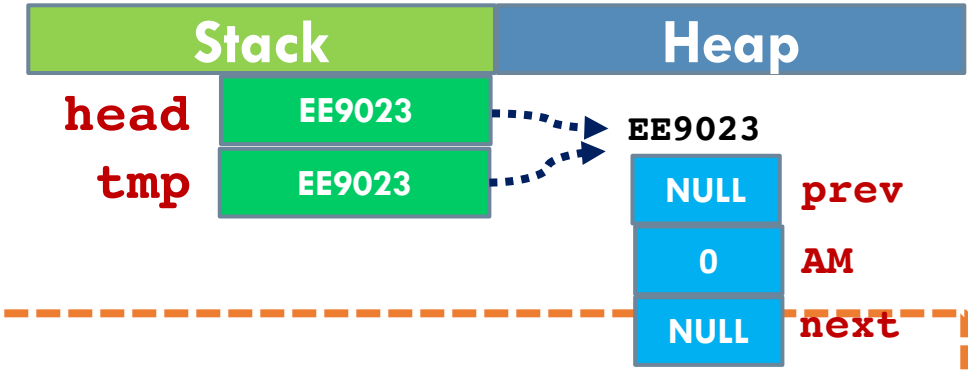

```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
          if(i==0){
             head=create_student_node(i);
             tmp=head;
          }
          else{
              tmp->next=create_student_node(i);
              tmp->next->prev=tmp;
              tmp=tmp->next;
          }
    }
    return head;
}
```



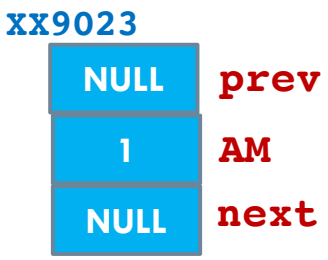

```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```

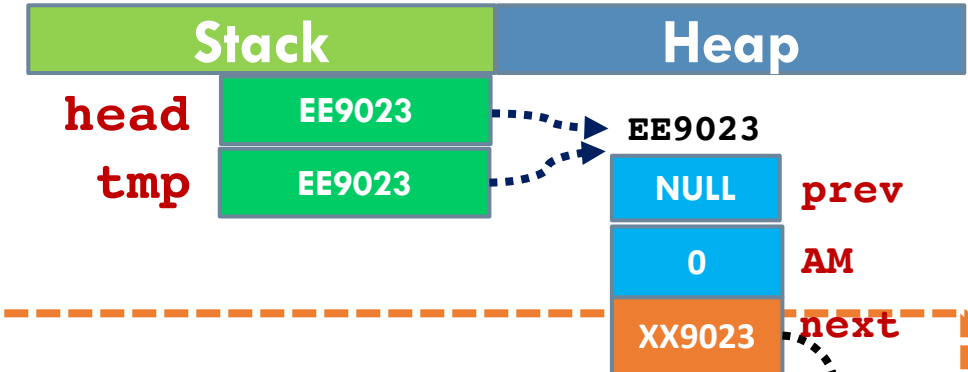



```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```

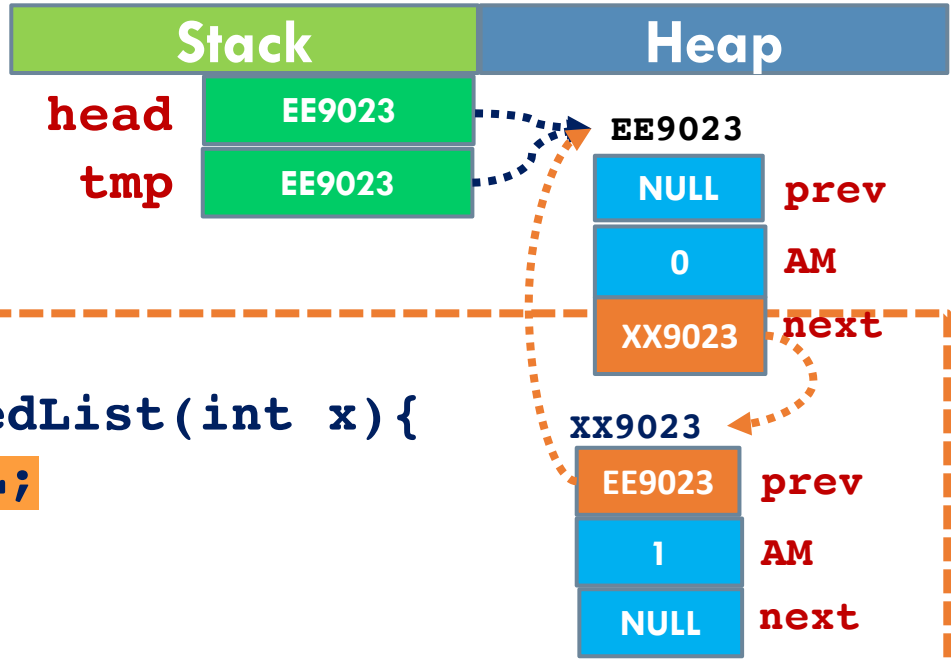



```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```

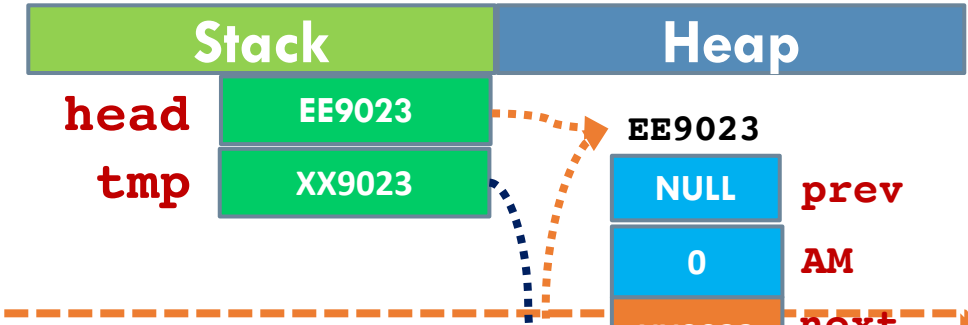




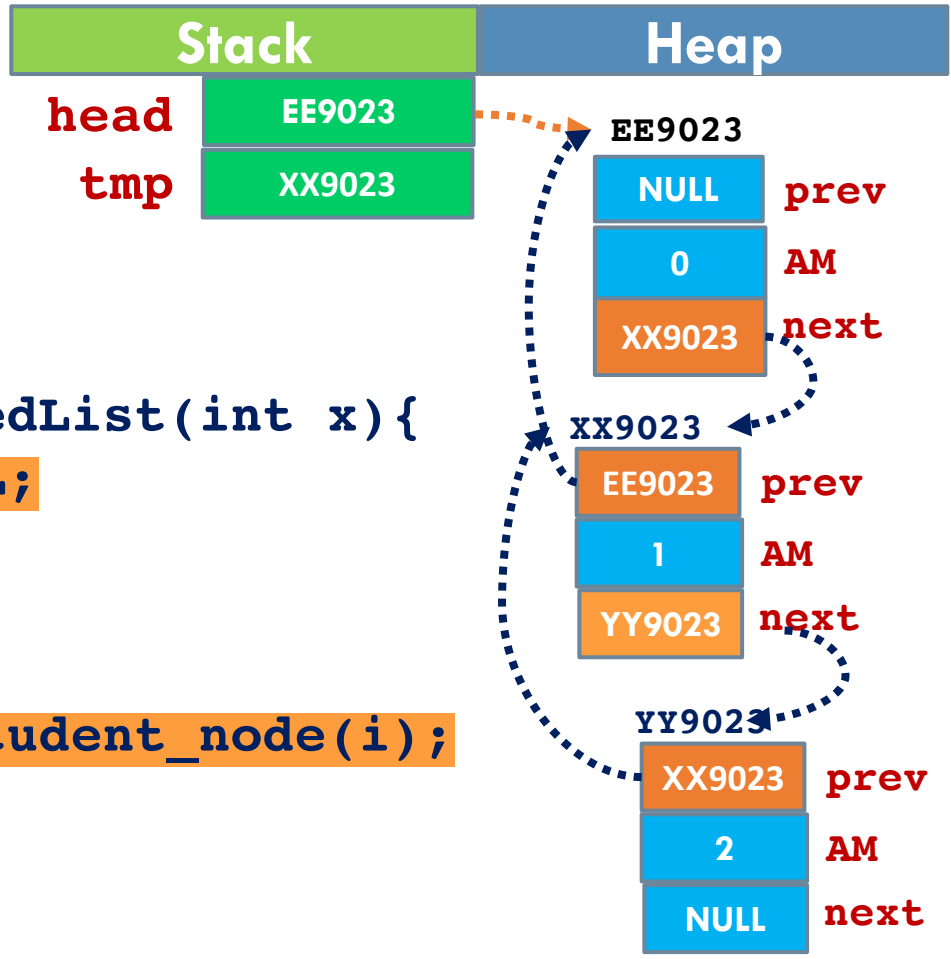

```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```




```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```




```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```




```
STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp=NULL;
    int i=0;
    for(;i<x;i++){
        if(i==0){
            head=create_student_node(i);
            tmp=head;
        }
        else{
            tmp->next=create_student_node(i);
            tmp->next->prev=tmp;
            tmp=tmp->next;
        }
    }
    return head;
}
```

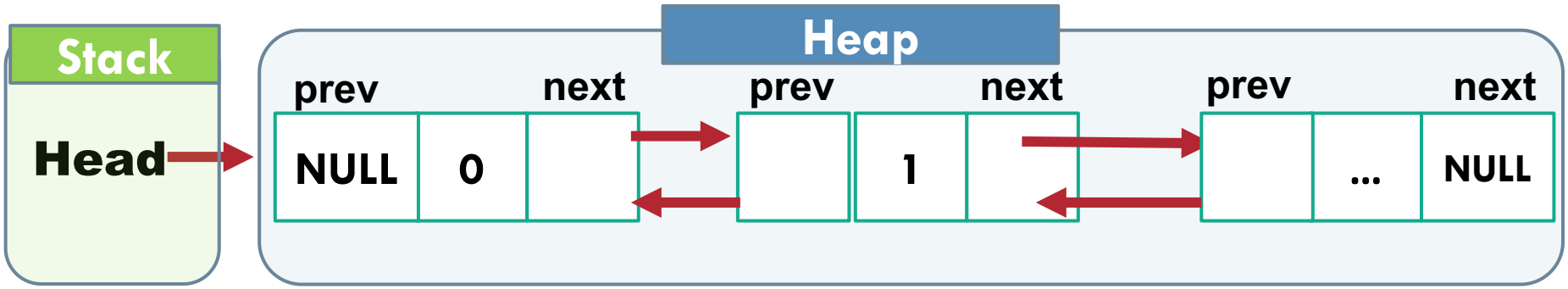
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
STUDENT * createDoubleLinkedList(int );
STUDENT* create_student_node(int AM);
void printlist(STUDENT *nd);
int main()
{
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    printlist(head);
    return 0;
}

STUDENT * createDoubleLinkedList(int x){
    STUDENT *head,*tmp;
    int i=0;
    for(;i<x;i++){
          if(i==0){
             head=create_student_node(i);
             tmp=head;
          }
          else{
              tmp->next=create_student_node(i);
              tmp->next->prev=tmp;
              tmp=tmp->next;
          }
    }
    return head;
}
void printlist(STUDENT *nd){
     STUDENT *  iterator;
     for(iterator=nd; iterator!=NULL; iterator = iterator->next){
             printf("student id: %d  %p   prev:%p  next:%p \n",  iterator->AM, iterator,
iterator->prev,iterator->next);
    }
}
```

```c
typedef struct node
{
    int AM;
    struct node *prev;
    struct node *next;
}  STUDENT;
```

```c
STUDENT* create_student_node(int AM) {
    STUDENT *node;
    node = (STUDENT *)malloc(sizeof(STUDENT));
    if (node == NULL) {
        printf("Memory not allocated.\n");
        exit(0);
    }
    else{
      node->prev = NULL;
      node->AM = AM;
      node->next = NULL;
      return node;
    }
}
```

```
student id: 0  47C058B0   prev:0  next:47C058D0
student id: 1  47C058D0   prev:47C058B0  next:47C058F0
student id: 2  47C058F0   prev:47C058D0  next:47C05910
student id: 3  47C05910   prev:47C058F0  next:47C05930
student id: 4  47C05930   prev:47C05910  next:47C05950
student id: 5  47C05950   prev:47C05930  next:47C05970
student id: 6  47C05970   prev:47C05950  next:47C05990
student id: 7  47C05990   prev:47C05970  next:47C059B0
student id: 8  47C059B0   prev:47C05990  next:47C059D0
student id: 9  47C059D0   prev:47C059B0  next:0
```

Αναζήτηση Κόμβου σε Λίστα

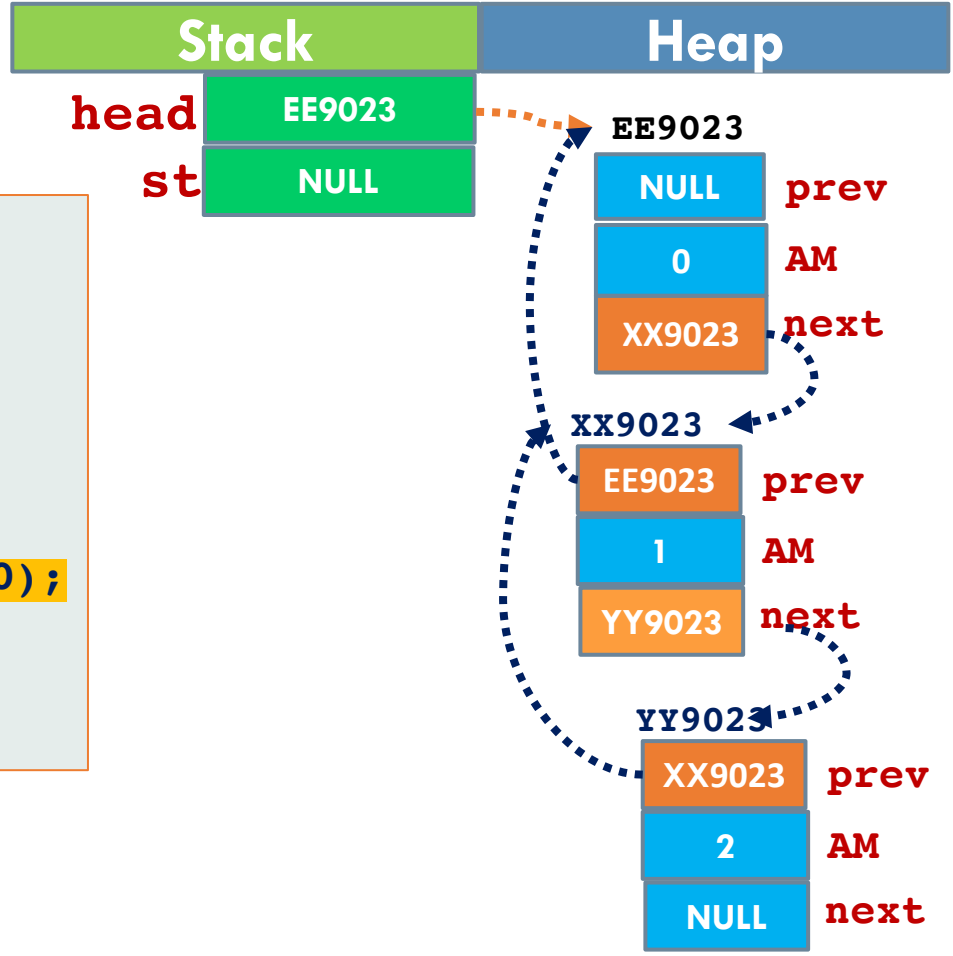



```
STUDENT*  getStudent(STUDENT *nd, int AM){
      STUDENT*  iterator;
      for(iterator=nd; iterator!=NULL; iterator = iterator->next){
              if (iterator->AM==AM) return iterator;
    }
    return NULL;
}
```

Αναζήτηση Κόμβου

Έστω ότι θέλω να αναζητήσω τον κόμβο με ΑΜ=1

```
STUDENT*  findStudent(STUDENT *nd, int AM);
STUDENT * createDoubleLinkedList(int );
STUDENT* create_student_node(int AM);
void printlist(STUDENT *nd);

int main(){
    STUDENT *head=NULL;
    STUDENT *st=NULL;
    head=createDoubleLinkedList(10);
    st=findStudent(head, 1);
return 0;
}
```

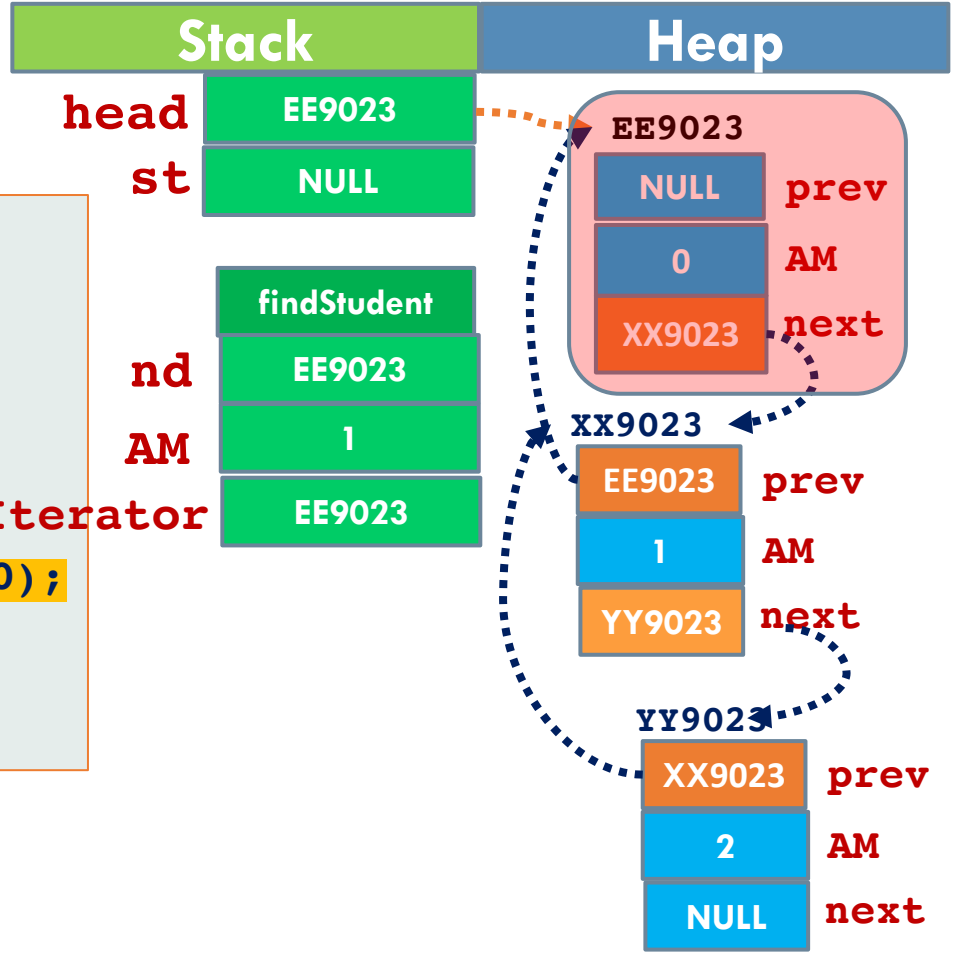



```
STUDENT*  findStudent(STUDENT *nd, int AM){
     STUDENT*  iterator;
     for(iterator=nd; iterator!=NULL; iterator = iterator->next){
            if (iterator->AM==AM) return iterator;
    }
    return NULL;
}
```

Αναζήτηση Κόμβου

Έστω ότι θέλω να αναζητήσω τον κόμβο με ΑΜ=1

```
STUDENT*  findStudent(STUDENT *nd, int AM);
STUDENT * createDoubleLinkedList(int );
STUDENT* create_student_node(int AM);
void printlist(STUDENT *nd);

int main(){
    STUDENT *head=NULL;
    STUDENT *st=NULL;
    head=createDoubleLinkedList(10);
    st=findStudent(head, 1);
return 0;
}
```

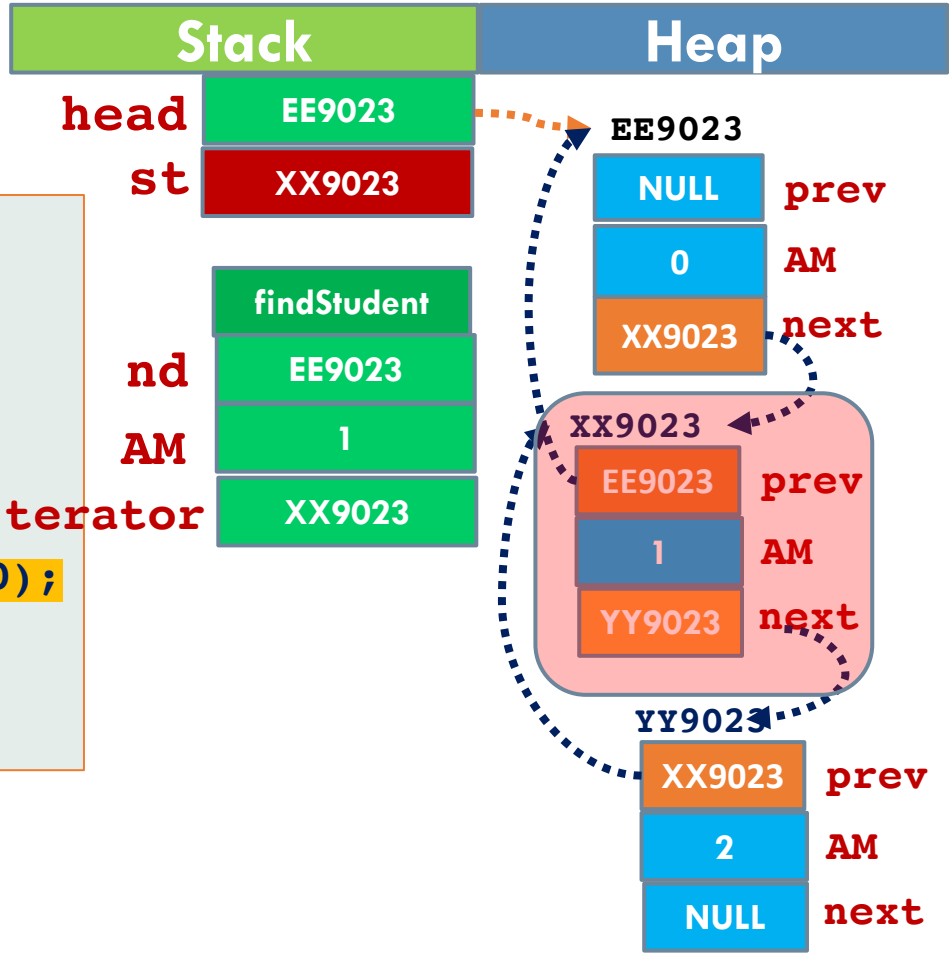



```
STUDENT*  findStudent(STUDENT *nd, int AM){
    STUDENT*  iterator;
    for(iterator=nd; iterator!=NULL; iterator = iterator->next){
            if (iterator->AM==AM) return iterator;
    }
    return NULL;
}
```

Αναζήτηση Κόμβου

Έστω ότι θέλω να αναζητήσω τον κόμβο με ΑΜ=1

```
STUDENT*  findStudent(STUDENT *nd, int AM);
STUDENT * createDoubleLinkedList(int );
STUDENT* create_student_node(int AM);
void printlist(STUDENT *nd);

int main(){
    STUDENT *head=NULL;
    STUDENT *st=NULL;
    head=createDoubleLinkedList(10);
    st=findStudent(head, 1);
return 0;
}
```

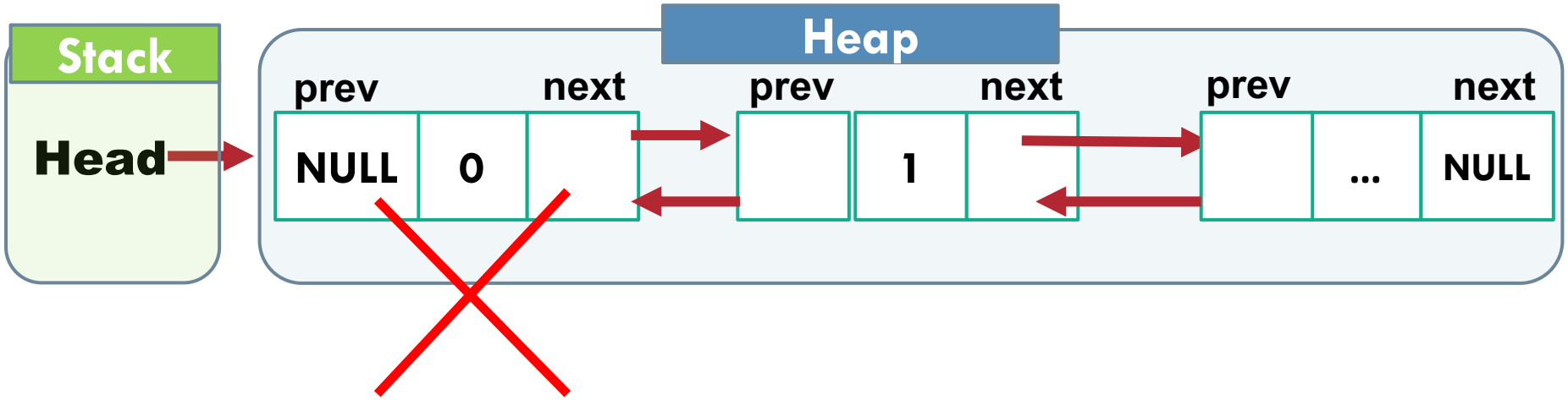



```
STUDENT*  findStudent(STUDENT *nd, int AM){
    STUDENT*  iterator;
    for(iterator=nd; iterator!=NULL; iterator = iterator->next){
            if (iterator->AM==AM) return iterator;
    }
    return NULL;
}
```

Διαγραφή ενός Κόμβου από μια Λίστα

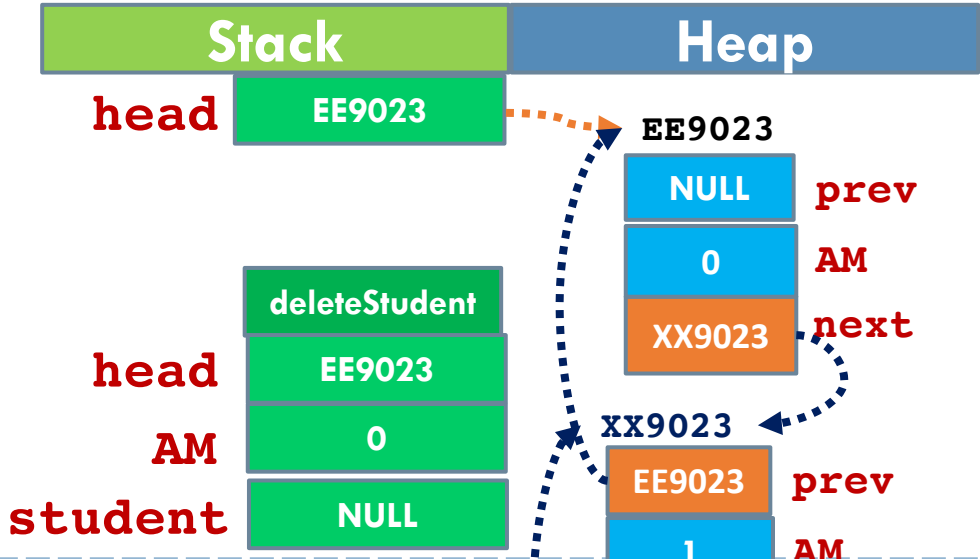
```
typedef struct node {
                int AM;
                struct node *prev;
                struct node *next;
} STUDENT;
```

- Τι πρέπει να κάνω αν θέλω να διαγράψω τον πρώτο κόμβο;



Έστω ότι θέλω να διαγράψω τον κόμβο με AM=0

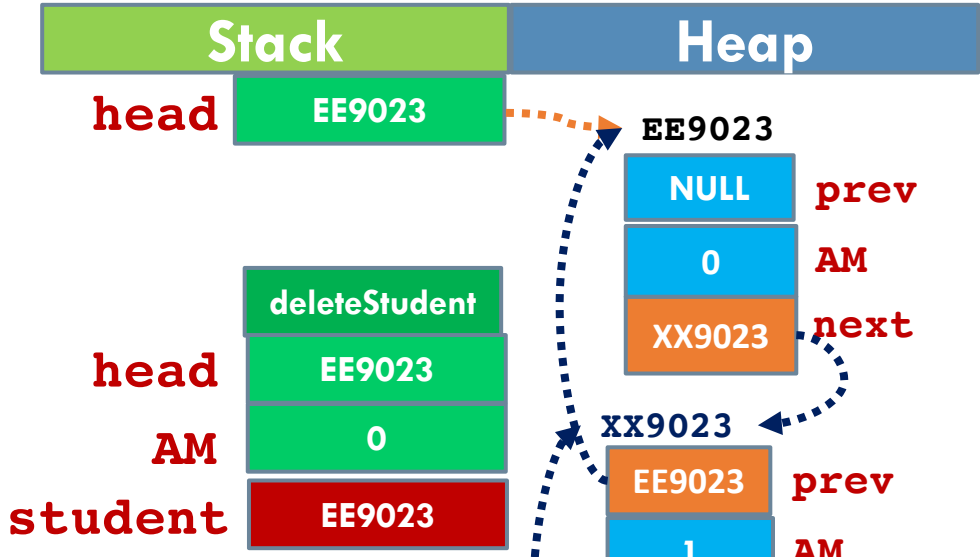
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=0

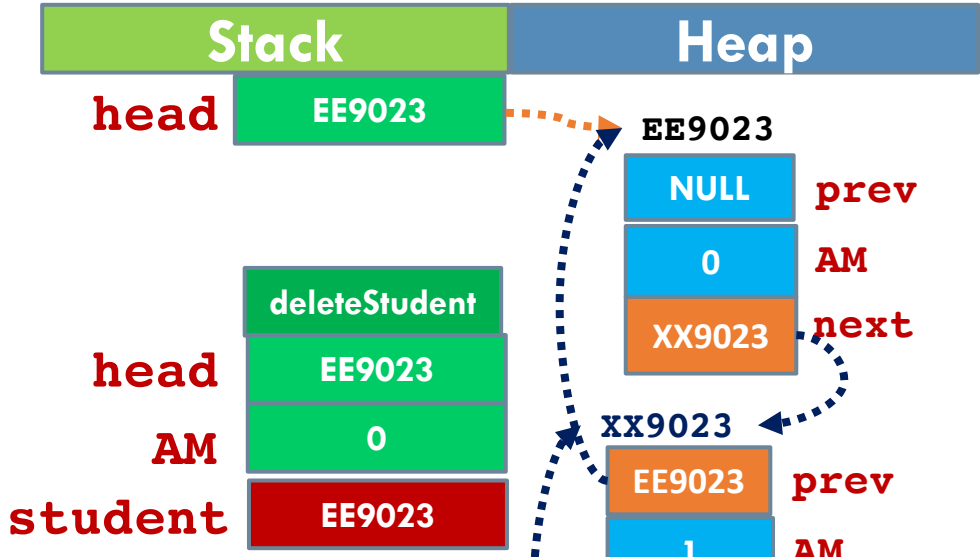
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=0

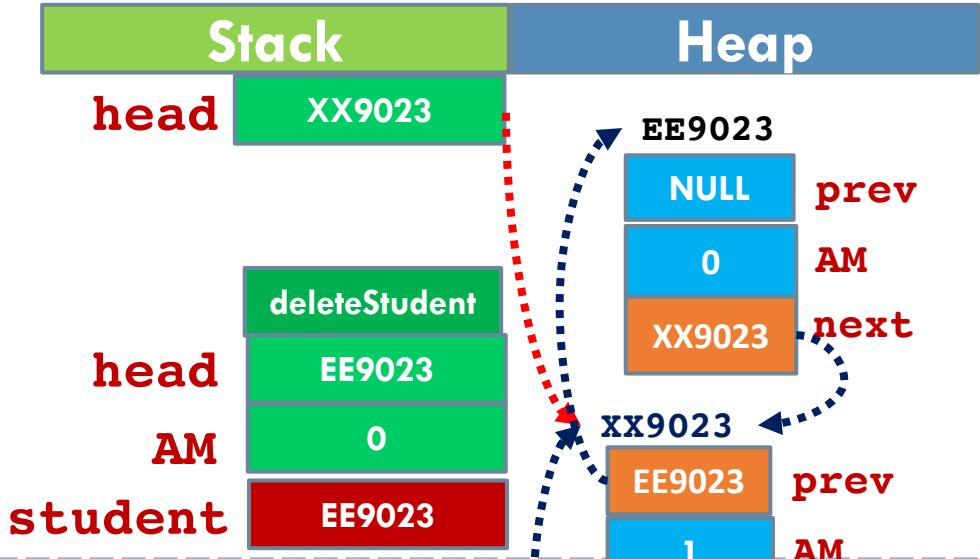
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=0

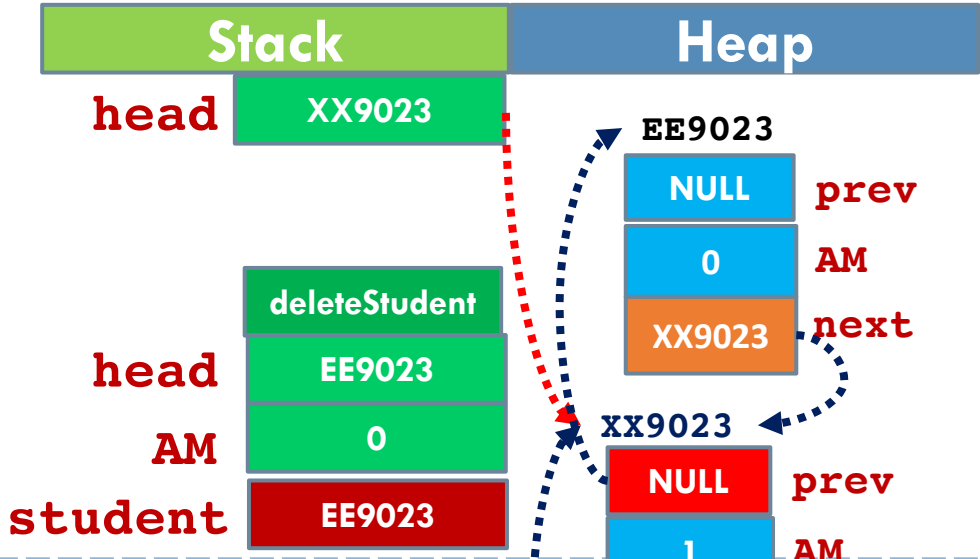
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                     student->prev->next=student->next;
                     student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=0

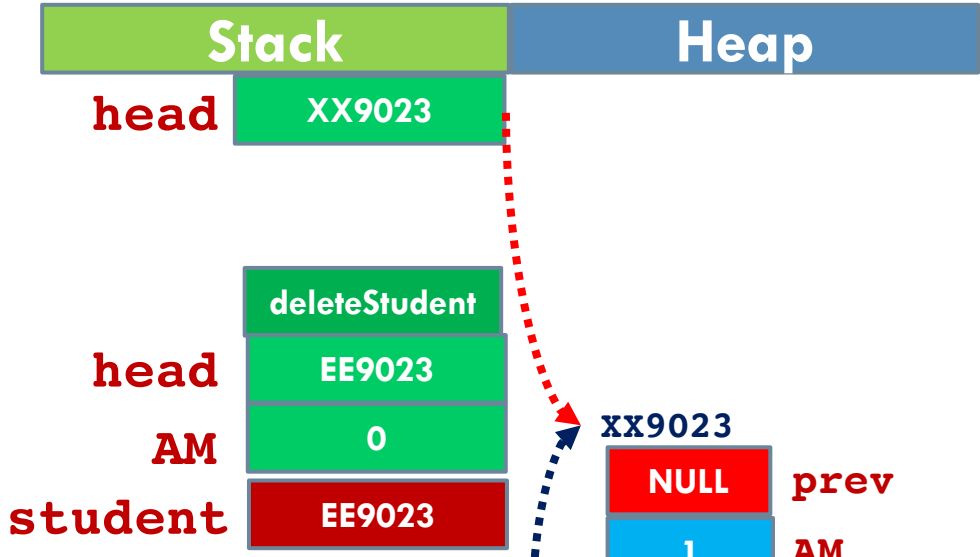
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                    student->prev->next=student->next;
                    student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=0

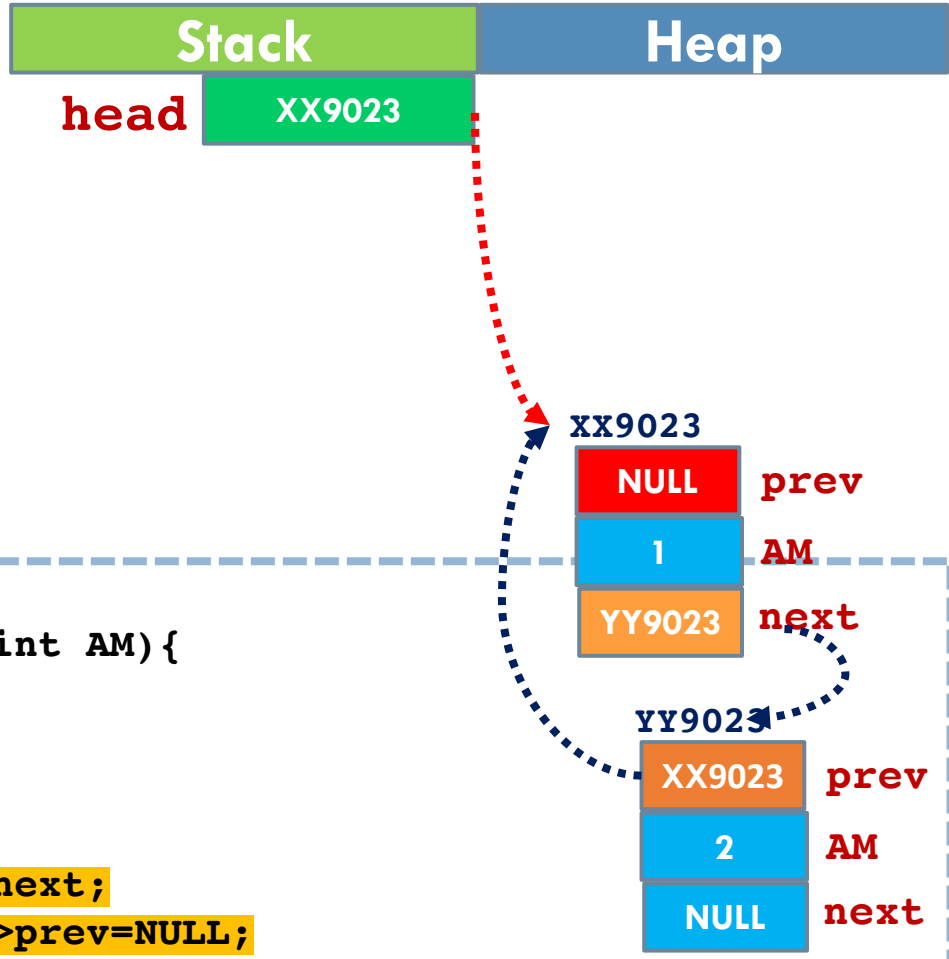
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
    return;
```

Έστω ότι θέλω να διαγράψω τον κόμβο με ΑΜ=0

```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,0);
return 0;
}
```

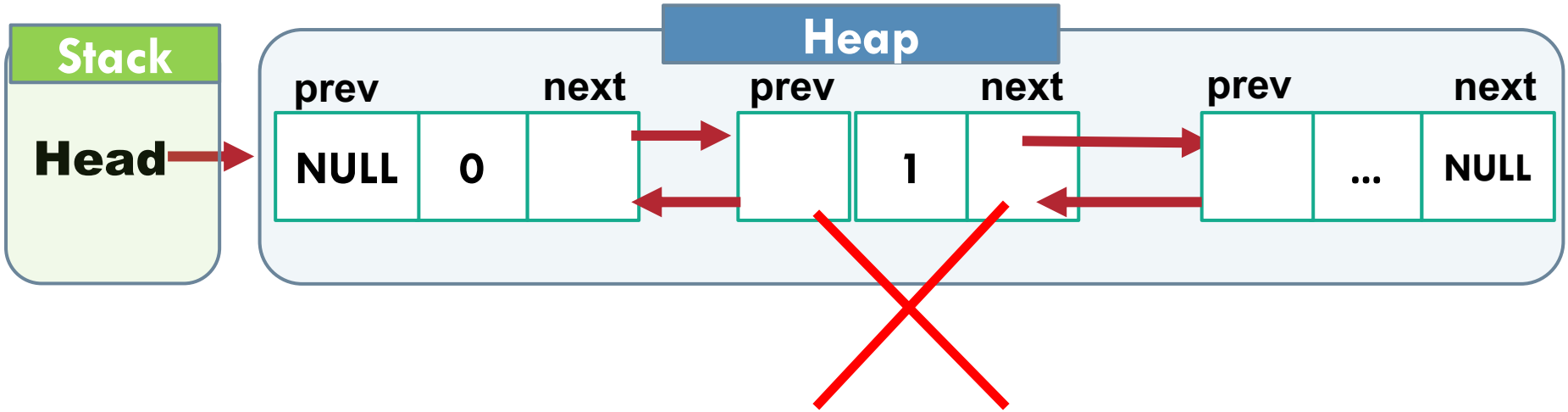



```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
    return;
```

Διαγραφή ενός Κόμβου από μια Λίστα

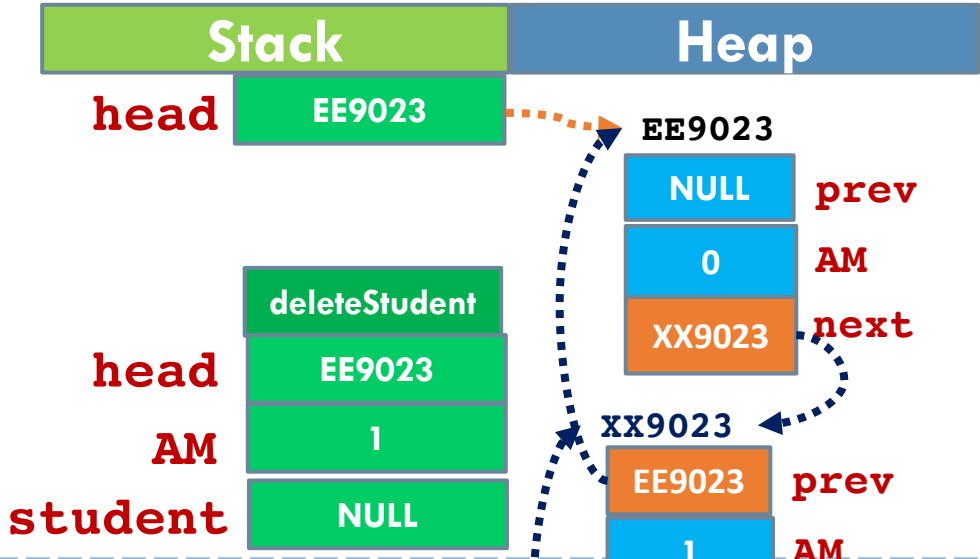
```
typedef struct node {
        int AM;
        struct node *prev;
        struct node *next;
} STUDENT;
```

- Τι πρέπει να κάνω αν θέλω να διαγράψω έναν ενδιάμεσο κόμβο;



Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

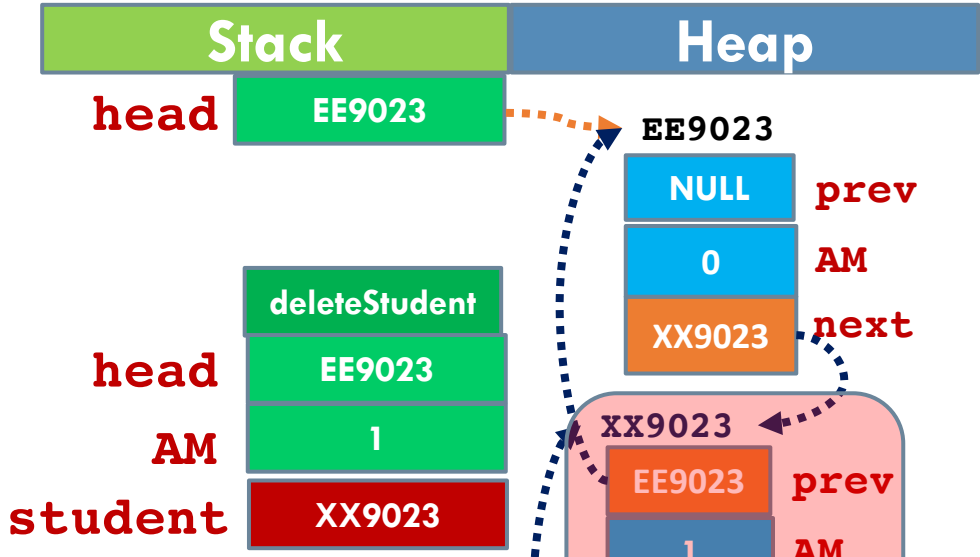
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

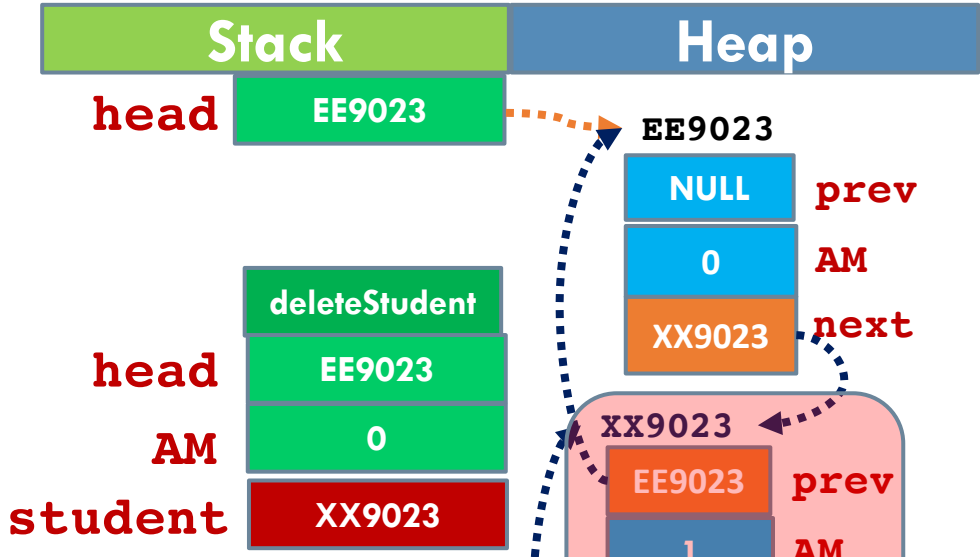
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

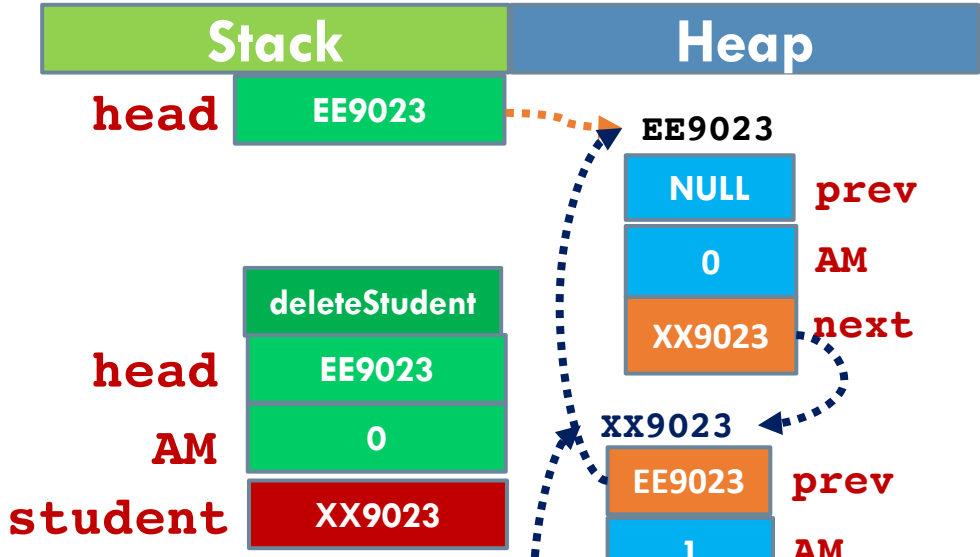
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

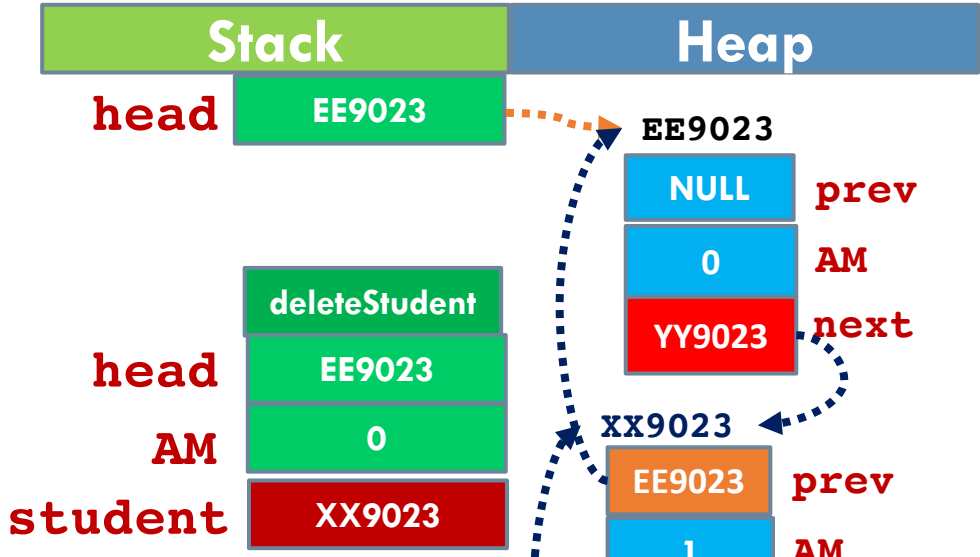
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                        student->prev->next=student->next;
                        student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

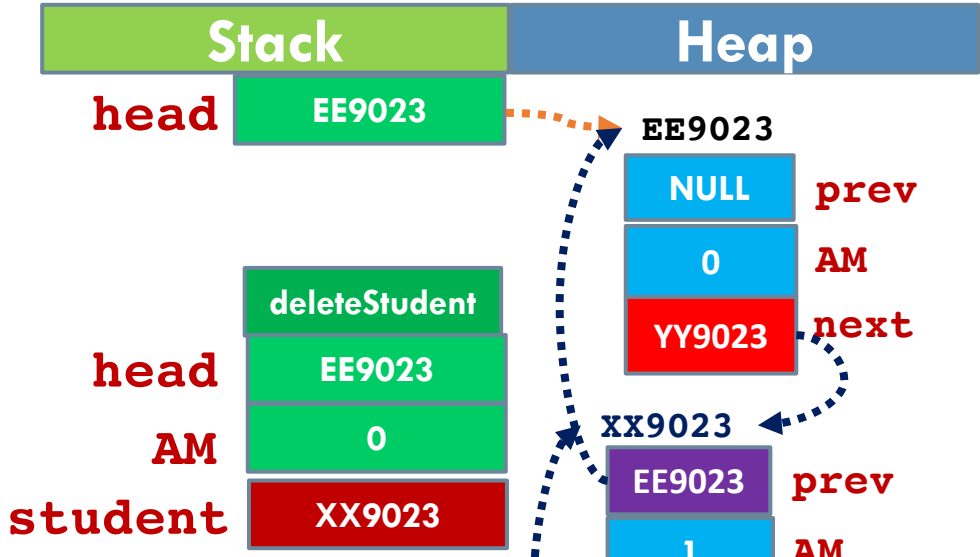
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                    student->prev->next=student->next;
                    student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

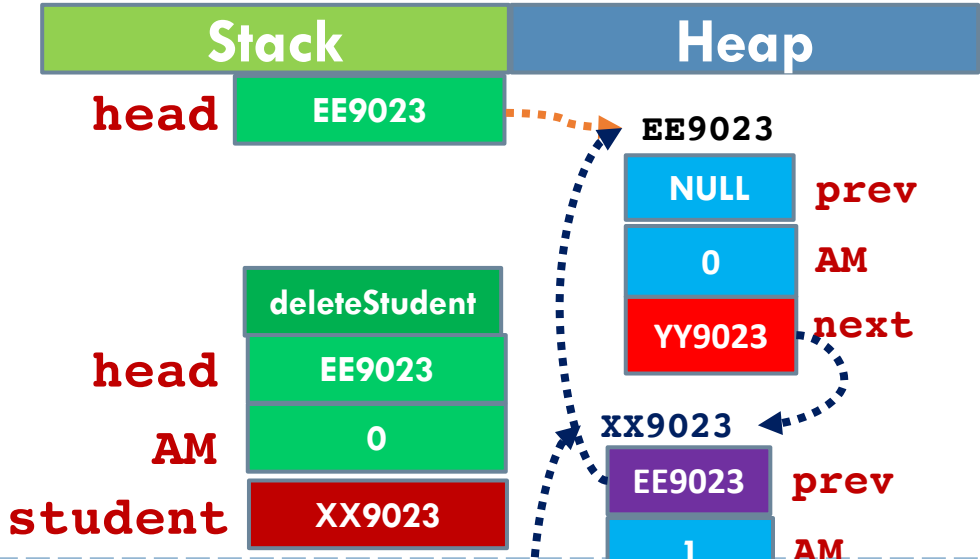
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                    student->prev->next=student->next;
                    student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

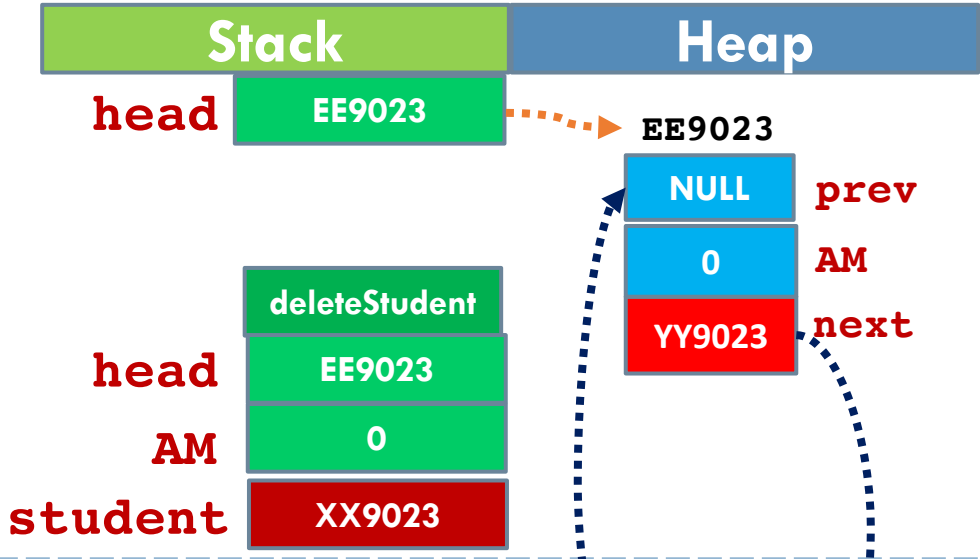
```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                    student->prev->next=student->next;
                    student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Έστω ότι θέλω να διαγράψω τον κόμβο με AM=1

```
int main(){
    STUDENT *head=NULL;
    head=createDoubleLinkedList(10);
    deleteStudent(&head,1);
return 0;
}
```




```
void deleteStudent(STUDENT **head, int AM){
    STUDENT *student=NULL;
    student=findStudent(*head, AM);
    if(student!=NULL){
            if(*head==student) {
                    *head=student->next;
                     student->next->prev=NULL;
            }
            else{

                            student->prev->next=student->next;
                            student->next->prev=student->prev;
            }
            printf("Deleted studend node: %d \n",student->AM);
            free(student);
    }
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/

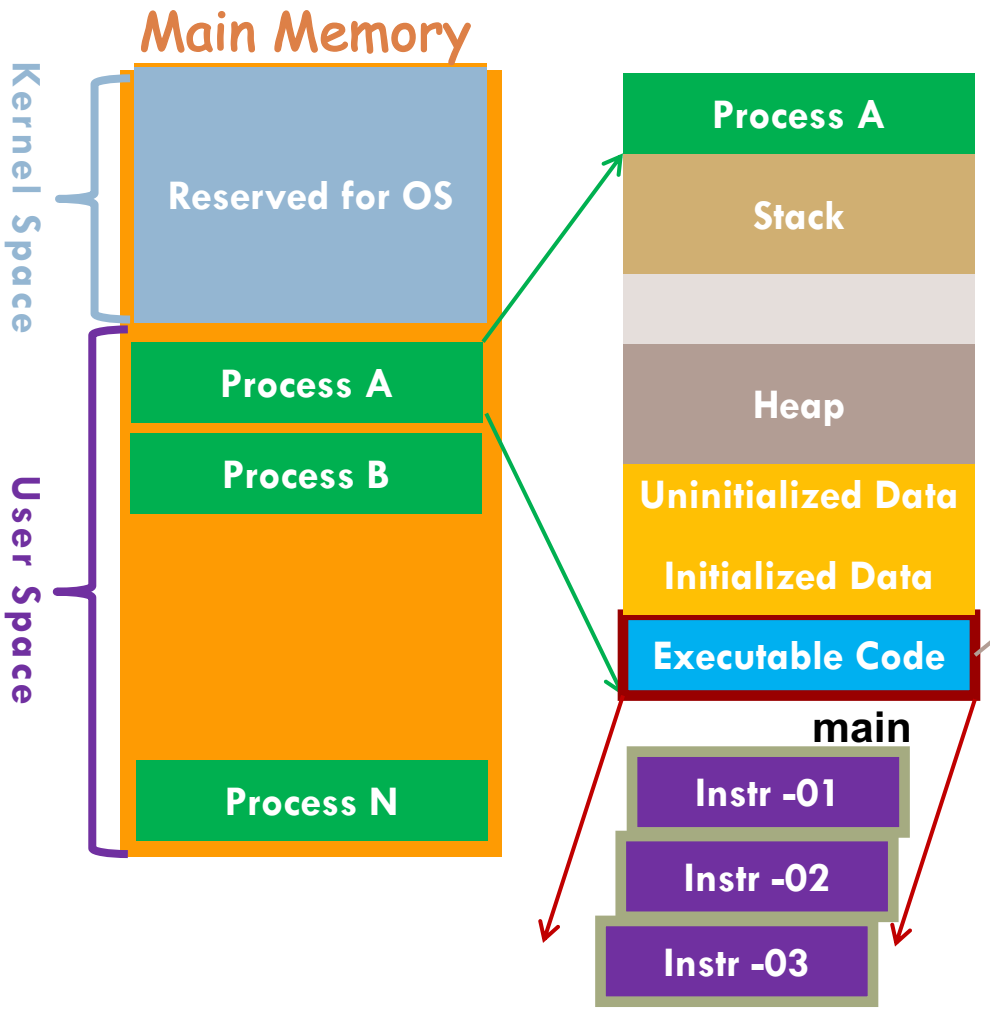
ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

10η Εβδομάδα: Διπλά Διασυνδεδεμένες Λίστες και Δείκτες σε Συνάρτηση

Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Διαχείριση μνήμης – Συναρτήσεις



Οι συναρτήσεις δεσμεύουν και αυτές μνήμη και άρα μπορούμε
να αναφερθούμε σε αυτές τις θέσεις μνήμης μέσα από τα προγράμματά μας.

Μπορούμε λοιπόν να χρησιμοποιούμε μεταβλητές που αποθηκεύουν την διεύθυνση της μνήμης στην οποία υπάρχει μια συνάρτηση.

Δείκτης σε Συνάρτηση

- Ένας δείκτης συνάρτησης (function pointer) είναι μια μεταβλητή τύπου δείκτη που δείχνει το σημείο στην μνήμη το οποίο περιέχει εκτελέσιμο κώδικα και όχι δεδομένα.

- Για παράδειγμα στον παρακάτω κώδικα ορίζω έναν δείκτη συνάρτησης function_p ο οποίος δυνητικά θα δείχνει σε εκτελέσιμο κώδικα συνάρτησης που θα δέχεται ως ορίσματα δυο ακέραιες μεταβλητές και θα επιστρέφει έναν ακέραιο.

```
int (*function_p)  (int, int);
int (*function_p[3])  (int, int);
```

```
int sum (int, int);
int sub (int, int);
int compare (int, int);
```

Τι θα δώσει η sizeof(function_p) ?

Δείκτης σε Συνάρτηση - Παράδειγμα

```
#include <stdio.h>
#include <stdlib.h>
int sum (int, int);
int sub (int, int);
int compare (int, int);

int main()
{
    int (*function_p)  (int, int);
    function_p=sum;
    printf("%d\n", function_p(1,2));
    function_p=sub;
    printf("%d\n", function_p(1,2));
    function_p=compare;
    printf("%d\n", function_p(1,2));

    return 0;
}

int sum (int x, int y) {return x+y;}
int sub (int x, int y) {return x-y;}
int compare (int x, int y) { return x>y?1:0;}
```

Δείκτης σε Συνάρτηση - Παράδειγμα

```c
#include <stdio.h>
#include <stdlib.h>
int sum (int, int);
int sub (int, int);
int compare (int, int);

int main()
{
    int (*function_p)  (int, int);
    function_p=&sum;
    printf("%d\n", function_p(1,2));
    function_p=&sub;
    printf("%d\n", function_p(1,2));
    function_p=&compare;
    printf("%d\n", function_p(1,2));

    return 0;
}

int sum (int x, int y) {return x+y;}
int sub (int x, int y) {return x-y;}
int compare (int x, int y) { return x>y?1:0;}
```

Οι παρακάτω εντολές είναι ισοδύναμες:

```c
function_p=sum;
function_p=&sum;
```

Πίνακας Δεικτών σε Συνάρτηση - Παράδειγμα

```c
#include <stdio.h>
#include <stdlib.h>
int sum (int, int);
int sub (int, int);
int compare (int, int);

int main()
{
    int (*function_p)  (int, int);
    function_p=&sum;
printf("%p %p %p\n", (void *) sum, (void *) function_p, (void *) &sum);
    printf("%d\n", function_p(1,2));
    function_p=&sub;
    printf("%d\n", function_p(1,2));
    function_p=&compare;
    printf("%d\n", function_p(1,2));

    return 0;
}

int sum (int x, int y) {return x+y;}
int sub (int x, int y) {return x-y;}
int compare (int x, int y) { return x>y?1:0;}
```

```
0x10c370f20 0x10c370f20 0x10c370f20
```

Δείκτης σε Συνάρτηση - Παράδειγμα

```c
#include <stdio.h>
#include <stdlib.h>
int sum (int, int);
int sub (int, int);
int compare (int, int);

int main()
{
    int (*function_p [3])  (int, int);
    function_p[0]=sum;
    function_p[1]=sub;
    function_p[2]=compare;

    printf("%d\n", function_p[0](1,2));
    printf("%d\n", function_p[1](1,2));
    printf("%d\n", function_p[2](1,2));
    return 0;
}

int sum (int x, int y) {return x+y;}
int sub (int x, int y) {return x-y;}
int compare (int x, int y) { return x>y?1:0;}
```

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdlib.h>
#include <stdio.h>
double one ( double );
double two ( double );

int main (void ) {
        double (*f[2])(double) = {one, two} ;
        double x = 3.47, y;
        int i ;

        do {
                printf("select function:\t");
                scanf("%d", &i);
                y = (*f[i-1])( x) ;
                printf("result %g\n", y);
        } while (1) ;
        return EXIT_SUCCESS;
}

double one(double x) { return (x + 1.0) ; }
double two(double x ) { return (x + 2.0) ; }
```

Αρχικοποίηση της δομής

Κλήση συνάρτησης βάση της επιλογής του χρήστη

Δείκτης σε συνάρτηση ως όρισμα

- Μπορούμε να χρησιμοποιούμε έναν δείκτη σε μια συνάρτηση ως όρισμα στα πλαίσια κλήσης μιας συνάρτησης.
- Για παράδειγμα το παρακάτω πρότυπο συνάρτησης έχει δυο ορίσματα έναν δείκτη σε αλφαριθμητικό *char** και έναν δείκτη σε συνάρτηση με πρότυπο *void (*)(char *)*.

```
void smart_print(char * , void (*)(char * ));
```

```c
#include <stdio.h>
#include <stdlib.h>
void print_mobile(char *);
void print_desktop(char *);
void print_smartwatch(char *);
void smart_print(char *s , void (*)(char * ));

int main()
{
    smart_print("Hello world", print_mobile);
    smart_print("Hello world", print_desktop);
    smart_print("Hello world", print_smartwatch);
    return 0;
}

void smart_print(char *s, void (*print)(char * string)){
     print(s);
}

void print_mobile(char *s){printf ("Mobile Device: %s\n",s);};
void print_desktop(char *s){printf ("Desktop Device: %s\n",s);};
void print_smartwatch(char *s){printf ("Smart Watch Device: %s\n",s);};
```

Πρότυπο της smart_print. Πόσα ορίσματα έχει και τι τύπου είναι το καθένα;

Κλήση της smart_print.

Υλοποίηση της smart_print.

Δήλωση Τύπων

- Μπορούμε για να απλοποιήσουμε τη σύνταξη των προγραμμάτων μας δηλώνοντας με χρήση της **typedef** τύπους *που αντιστοιχούν σε δείκτες συναρτήσεων*.
  - **Ο τύπος printCallBack αντιστοιχεί σε δείκτη συνάρτησης με πρότυπο void (*) (char *).**

```
typedef void (*printCallBack)(char *);
```

```c
#include <stdio.h>
#include <stdlib.h>
typedef void (*printCallBack)(char *);

void print_mobile(char *);
void print_desktop(char *);
void print_smartwatch(char *);
void smart_print(char *s , printCallBack);

int main()
{
    smart_print("Hello world", print_mobile);
    smart_print("Hello world", print_desktop);
    smart_print("Hello world", print_smartwatch);
    return 0;
}

void smart_print(char *s, printCallBack print){
      print(s);
}

void print_mobile(char *s){printf ("Mobile Device: %s\n",s);};
void print_desktop(char *s){printf ("Desktop Device: %s\n",s);};
void print_smartwatch(char *s){printf ("Smart Watch Device: %s\n",s);};
```

Απλοποίηση της σύνταξης

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>
int map (int a[], int size, int (*f)(int, int));
int more(int a, int b);
int less(int a, int b);

int main(void) {
         int data[7]= {-1, 2, 1, 0, 5, 7, -3};
         printf("max: %2d\n", map(data, 7, more));
         printf("max: %2d\n", map(data, 7, less));
 return 0;
}

 int less (int a, int b) {return a > b;}

 int more (int a, int b) {return a < b;}

 int map (int a[], int size, int (*f)(int, int)) {
   int i , temp;
   temp = a[0];
   for (i=1; i < size; i++)
          if ( f(temp, a[i])) temp = a[i];

   return temp;
 }
```

Χωρίς τη χρήση δείκτη σε συνάρτηση

```
int getmin (int a[], int size)
{
 int i , temp;
 temp = a[0];

 for (i=1; i < size; i++) {
   if ( temp>a[i] ) temp =
   a[i];
   }

 return temp;
}
```

```
int getmax (int a[], int
size) {
  int i , temp;
  temp = a[0];

  for (i=1; i < size; i++) {
    if ( temp<a[i] ) temp =
a[i];
    }

  return temp;
}
```

Ενσωμάτωση δείκτη σε συνάρτηση ως πεδίο σε δομή δεδομένων

- Μπορούμε να ορίσουμε πεδία μιας δομής δεδομένων έτσι ώστε να δείχνουν σε μια συνάρτηση.
- Με αυτόν τον τρόπο επεκτείνουμε τη δομή δεδομένων ενσωματώνοντας σε αυτή και λειτουργικά χαρακτηριστικά

```
typedef int (*PassFailCallBack)(float);
typedef void (*EnterGradeCallBack)(float *);

typedef struct node{
  float grade;
  EnterGradeCallBack set_grade_func;
  PassFailCallBack pf_func;
} STUDENT;
```

```c
#include <stdio.h>
#include <stdlib.h>
typedef int (*PassFailCallBack)(float );
typedef void (*EnterGradeCallBack)(float *);


typedef struct node{
    float grade;
    EnterGradeCallBack set_grade_func;
    PassFailCallBack pf_func;
} STUDENT;

int pass_or_fail (float x);
void set_grade (float *z) ;

int main()
{
    STUDENT x={0, set_grade, pass_or_fail};
    x.set_grade_func(&(x.grade));
    printf ("%d",x.pf_func(x.grade) );
    return 0;
}

int pass_or_fail (float x) { return x>=5?1:0;}
void set_grade (float *z) {
    printf("Enter grade:");
    scanf("%f", z);
    printf ("grade: %1.2f \n", *z );
    return;
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/