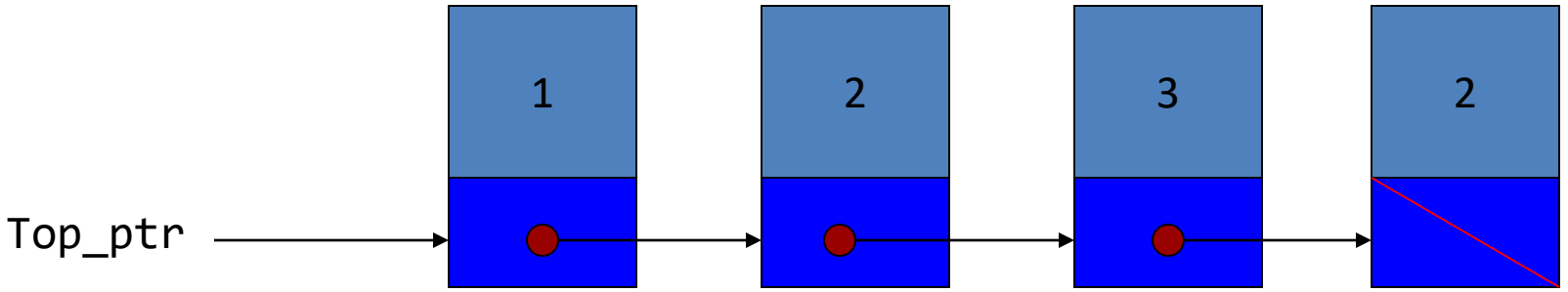
Διαδικαστικός Προγραμματισμός

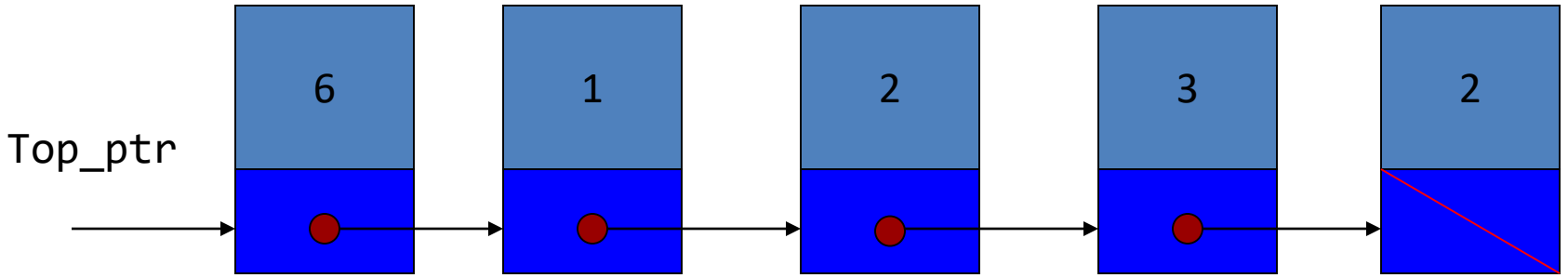
Βασίλης Παλιουράς

Στοίβες - stacks

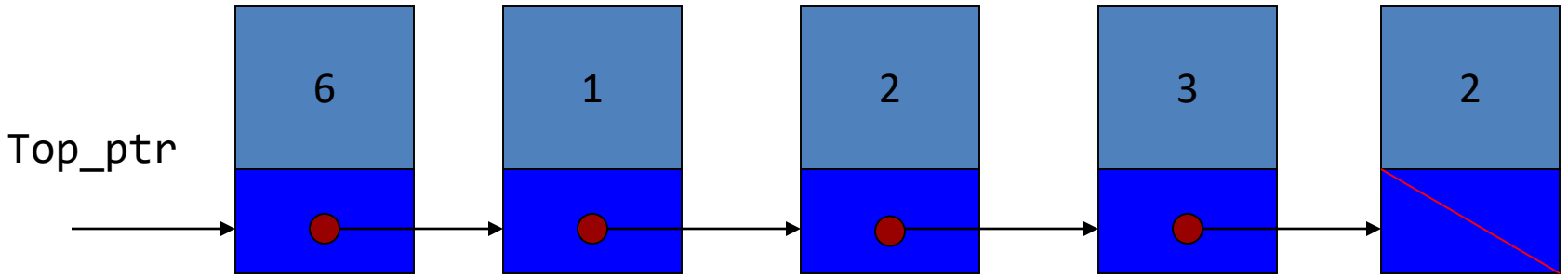


- Λειτουργίες push/pop
- Last-In First-Out (LIFO)

Στοίβες – stacks: push



Στοίβες – stacks: pop



Συμβολισμός προθέματος
Prefix notation

- `2*(5*(5 + 2) + 3*5)` `[= 100]`
- `2*(5*(+ 5 2) + *3 5)`
- `2*((* 5 + 5 2) + * 3 5)`
- `2*(+ * 5 + 5 2  * 3 5)`
- `  *2 +* 5 + 5 2 * 3 5`

- Διατηρώντας την προτεραιότητα, αντικαθιστώ κάθε έκφραση α ◊ β με ◊ α β
- Ο τελεστής προηγείται των τελεστέων
- Η τελική έκφραση *δε χρειάζεται παρενθέσεις!*

Περιεχόμενα stack σε κάθε βήμα:

* 2 + * 5 + 5 2 * 3 5

Διατρέχω την έκφραση από
δεξιά προς ***αριστερά***.

όταν βρίσκω αριθμό
    τον τοποθετώ στο σωρό
όταν βρίσκω τελεστή, τον εφαρμόζω
    στην κορυφή του σωρού
    και τοποθετώ το αποτέλεσμα
    στο σωρό

5

3 5

15

2 15

5 2 15

7 15

5 7 15

35 15

50

2 50

100

```c
#include <stdio.h>
#include <stdlib.h>

struct stackNode {/* self-referential
                     structure */
   int data;
   struct stackNode *nextPtr;
};

typedef struct stackNode StackNode;
typedef StackNode *StackNodePtr;

void push( StackNodePtr *, int );
int pop( StackNodePtr * );
int isEmpty( StackNodePtr );
void printStack( StackNodePtr );
void instructions( void );
```

```c
int main() {
   StackNodePtr stackPtr = NULL;  /* points to stack top */
   int choice, value;

   instructions();
   printf( "? " );
   scanf( "%d", &choice );

   while ( choice != 3 ) {

      switch ( choice ) {
         case 1:      /* push value onto stack */
            printf( "Enter an integer: " );
            scanf( "%d", &value );
            push( &stackPtr, value );
            printStack( stackPtr );
            break;
         case 2:      /* pop value off stack */
            if ( !isEmpty( stackPtr ) )
               printf( "The popped value is %d.\n",
                       pop( &stackPtr ) );

            printStack( stackPtr );
            break;
         default:
            printf( "Invalid choice.\n\n" );
            instructions();
            break;
      }

      printf( "? " );
      scanf( "%d", &choice );
   }

   printf( "End of run.\n" );
   return 0;
}
```

```c
/* Print the instructions */
void instructions( void )
{
   printf( "Enter choice:\n"
           "1 to push a value on the stack\n"
           "2 to pop a value off the stack\n"
           "3 to end program\n" );
}

/* Insert a node at the stack top */
void push( StackNodePtr *topPtr, int info )
{
   StackNodePtr newPtr;

   newPtr = malloc( sizeof ( StackNode ) );
   if ( newPtr != NULL ) {
      newPtr->data = info;
      newPtr->nextPtr = *topPtr;
      *topPtr = newPtr;
   }
   else
      printf( "%d not inserted. No memory available.\n",
              info );
}
```

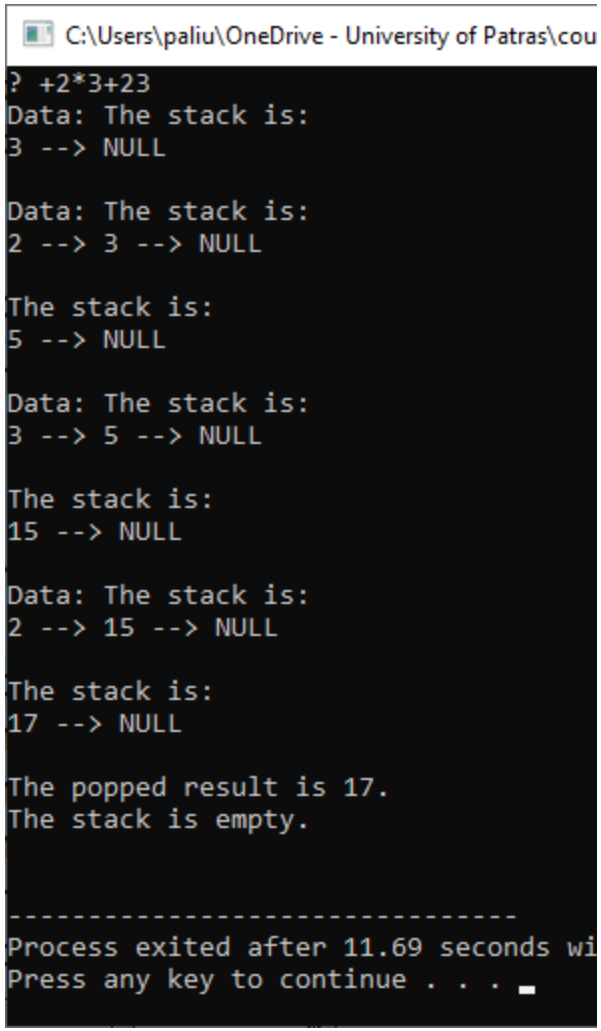
```c
/* Print the stack */
void printStack( StackNodePtr currentPtr )
{
   if ( currentPtr == NULL )
      printf( "The stack is empty.\n\n" );
   else {
      printf( "The stack is:\n" );

      while ( currentPtr != NULL ) {
         printf( "%d --> ", currentPtr->data );
         currentPtr = currentPtr->nextPtr;
      }

      printf( "NULL\n\n" );
   }
}
```

```c
/* Is the stack empty? */
int isEmpty( StackNodePtr topPtr )
{
   return topPtr == NULL;
}

/* Remove a node from the stack top */
int pop( StackNodePtr *topPtr )
{
   StackNodePtr tempPtr;
   int popValue;

   tempPtr = *topPtr;
   popValue = ( *topPtr )->data;
   *topPtr = ( *topPtr )->nextPtr;
   free( tempPtr );

   return popValue;
}
```

Παράδειγμα: Υπολογισμός έκφρασης προθέματος

- Χρησιμοποιώντας ένα stack, να γράψετε ένα πρόγραμμα που να υπολογίζει μια έκφραση προθέματος.
- *Προσχέδιο* λύσης
  - Υλοποιεί τον αλγόριθμο της διαφάνειας 6
  - Η είσοδος είναι ένα αλφαριθμητικό χωρίς κενά, με μονοψήφια όχι προσημασμένα δεδομένα, χωρίς ελέγχους,…

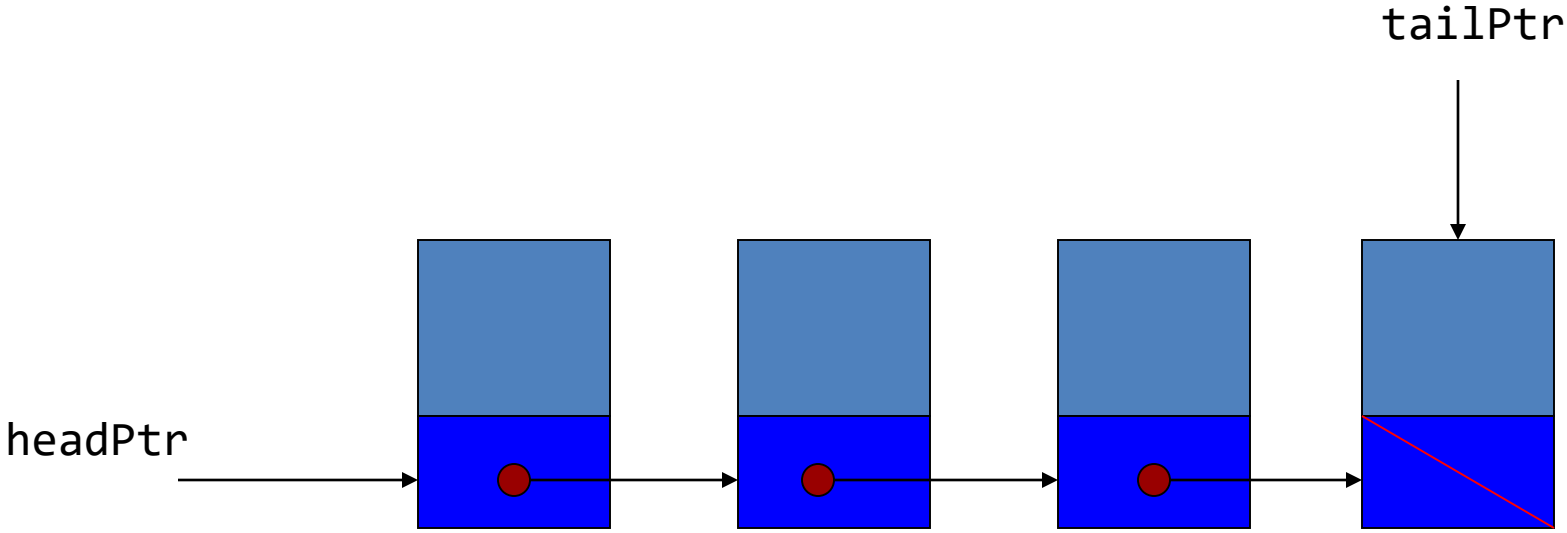
```c
int main(void)
{
   StackNodePtr stackPtr = NULL;  /* points to stack top */
   int choice, value, i;
   int a, b;
   char expression[100];
   printf( "? " );
   scanf( "%99s", expression ); /* Can you explain %99s ? */

   for(i=strlen(expression)-1; i>=0; i--) {    /* read from right to left */
               choice = expression[i];
               if (isdigit(choice)) {          /* a single char only!!! */
                         value = choice - 48; /* Can you explain why? */
                         /* push data value onto stack */
                         printf("Data: ");
                         push( &stackPtr, value );
                         printStack( stackPtr );
                         }
                else {
                         a = pop(&stackPtr);
                         b = pop(&stackPtr);
                         switch(choice) {
                                   case '+': push(&stackPtr, a + b);
                                             printStack( stackPtr );
                                             break;
                                   case '*': push(&stackPtr, a * b);
                                             printStack( stackPtr );
                                             break;
                         }
                 }
   }
   /* pop result value off stack */
   if ( !isEmpty( stackPtr ) )
        printf( "The popped result is %d.\n", pop( &stackPtr ) );

   printStack( stackPtr );

   return EXIT_SUCCESS;
}
```



Για το σπίτι – Προαιρετική Άσκηση 2

- Να εισαγάγετε κατάλληλους ελέγχους ώστε το πρόγραμμα να χειρίζεται και μη έγκυρη είσοδο, τυπώνοντας μήνυμα λάθους.
- Να τροποποιήσετε τον κώδικα ώστε να χειρίζεται
  - πολυψήφιους μη αρνητικούς αριθμούς
  - που διαχωρίζονται με κενά
- Να τροποποιήσετε τον κώδικα ώστε να χειρίζεται και μη αντιμεταθετικούς τελεστές (-, /)
- Να τροποποιήσετε τον κώδικα ώστε να χειρίζεται αριθμούς με υποδιαστολή (ως υποδιαστολή, εννοείται η τελεία)
- Η είσοδος θα πρέπει να είναι πάντα **ένα μόνο αλφαριθμητικό για το σύνολο της έκφρασης.**
  - **Χρησιμοποιήστε την fgets () με stream το stdin.**
  - **Αξιολογήστε τη χρήση συναρτήσεων atoi, strtok, strcspn, …**
  - **Μεριμνήστε να μην υπάρχει περίπτωση buffer overflow στο πρόγραμμά σας.**
  - **Μεριμνήστε να μην δημιουργούνται memory leaks στο πρόγραμμά σας.**
  - **Στο eclass μέχρι 31 Μαΐου, κάθε bullet σε ξεχωριστό project, όλα ένα zip.**
  - **Θυμηθείτε -Wall, -Wextra, -std=c90, -pedantic, cppcheck, valgrind, drmemory,…**

Ουρές - queues



- Λειτουργίες enqueue/dequeuer
- First-In First-Out (FIFO)

Enqueue/Dequeue

- Enqueue: Στοιχεία τοποθετούνται στο tail

```
(head) A → B → C → NULL          (tail) ήρθε C
(head) A → B → C → D → NULL      (tail) ήρθε D
(head) A → B → C → D → E → NULL (tail) ήρθε E
```

- Dequeue: Στοιχεία αποχωρούν από head

```
εξυπηρετείται το A   (head)     B → C → D → E → NULL   (tail)
εξυπηρετείται το B   (head)         C → D → E → NULL   (tail)
```

```c
#include <stdio.h>
#include <stdlib.h>

struct queueNode {   /* self-referential structure */
   char data;
   struct queueNode *nextPtr;
};

typedef struct queueNode QueueNode;
typedef QueueNode *QueueNodePtr;

/* function prototypes */
void printQueue( QueueNodePtr );
int isEmpty( QueueNodePtr );
char dequeue( QueueNodePtr *, QueueNodePtr * );
void enqueue( QueueNodePtr *, QueueNodePtr *, char );
void instructions( void );
```

```c
void instructions( void )
{
   printf ( "Enter your choice:\n"
            "   1 to add an item to the queue\n"
            "   2 to remove an item from the queue\n"
            "   3 to end\n" );
}
```

```c
int main(void) {
   QueueNodePtr headPtr = NULL,
                tailPtr = NULL;
   int choice;
   char item;

   instructions();
   printf( "? " );
   scanf( "%d", &choice );

   while ( choice != 3 ) {
      switch( choice ) {
       case 1:
         printf( "Enter a character: " );
         scanf( "\n%c", &item );
            enqueue( &headPtr, &tailPtr, item);
            printQueue( headPtr );
            break;

case 2:
        if ( !isEmpty( headPtr ) ) {
          item = dequeue( &headPtr, &tailPtr );
          printf( "%c has been dequeued.\n", item );
          }
         printQueue( headPtr );
          break;
default:
          printf( "Invalid choice.\n\n" );
          instructions();
          break;
      }

      printf( "? " );
      scanf( "%d", &choice );
   }
   printf( "End of run.\n" );
   return 0;
}
```

```c
void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr, char value )
{
   QueueNodePtr newPtr;

   newPtr = malloc( sizeof( QueueNode ) );

   if ( newPtr != NULL ) {
      newPtr->data = value;
      newPtr->nextPtr = NULL;

      if ( isEmpty( *headPtr ) )
         *headPtr = newPtr;
      else
         ( *tailPtr )->nextPtr = newPtr;

      *tailPtr = newPtr;
   }
   else
      printf( "%c not inserted. No memory available.\n", value );
   return ;
}
```

```
char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr )
{
   char value;
   QueueNodePtr tempPtr;

   value = ( *headPtr )->data;
   tempPtr = *headPtr;
   *headPtr = ( *headPtr )->nextPtr;

   if ( *headPtr == NULL )
      *tailPtr = NULL;

   free( tempPtr );
   return value;
}
```

```c
int isEmpty( QueueNodePtr headPtr )
{
   return headPtr == NULL;
}

void printQueue( QueueNodePtr currentPtr )
{
   if ( currentPtr == NULL )
      printf( "Queue is empty.\n\n" );
   else {
      printf( "The queue is:\n" );

      while ( currentPtr != NULL ) {
         printf( "%c --> ", currentPtr->data );
         currentPtr = currentPtr->nextPtr;
      }

      printf( "NULL\n\n" );
   }
}
```