Implementácia údajovej štruktúry zásobník s využitím poľa

Znenie zadania:

Pridajte podporu operácie TopAndPop, ktorá prečíta a zároveň odoberie prvok z vrcholu zásobníka (nie len zavolaním existujúcich operácií).

Prototyp operácie TopAndPop (stacka.h):

```
TElement TopAndPop( Stack S );
```

Pridajte podporu operácie PrintStack, ktorá vypíše obsah zásobníka, bez jeho modifikácie.

Prototyp operácie PrintStack:

```
void PrintStack( Stack S );
```

Poznámka: Pre overenie činnosti pridaných operácií doplňte aj modul pre testovanie.

Opis riešenia:

Funkcia TopAndPop má načítať a následne po prečítaní odobrať prvok z vrcholu zásobníka. Najprv sa vytvorí pomocná premenná typu TElement. Do nej sa uloží hodnota vrchného prvku. Táto hodnota sa vypíše a posunie sa vrchol zásobníka, čím sa zo stacku vyberie prvok.

Funkcia PrintStack má vypísať zásobník. Opäť sa najprv vytvorí pomocná premenná, ktorá bude odkazovať na prvok ktorý sa práve vypisuje. Začína sa vypísaním najvrchnejšieho prvku čiže Temp sa inicialuzuje na hodnotu adresy poľa kde je vrchný prvok. Nasleduje cyklus v ktorom sa vypisuje prvok poľa s indexom ktorý je uložený v Temp a Temp sa dekrementuje. Tento cyklus sa opakuje až po posledný prvok, čiže kým sa Temp nerovná 0.

Pre otestovanie funkčnosti funkcií som ešte upravil funkciu main. V nej sa po vytvorení a naplnení zásobníka číslami od 0 po 10 volajú postupne vytvorené funkcie. Najprv sa zavolá funkcia PrintStack, po nej funkcia TopAndPop. Výpis po oboch funkciách by mal byť rovnaký, ale po funkcii TopAndPop má zásobník ostať prázdny. To overujem volaním funkcie IsEmpty.

Zdrojový kód:

funkcia TElement TopAndPop(Stack S):

```
TElement TopAndPop ( Stack S ){
       if (!IsEmpty(S)){
           TElement Temp;
           Temp = S->Array[ S->TopOfStack ];
           printf("%d ",Temp);
           S->TopOfStack--;
           return Temp;
       }else{
           Error( "Stack is empty!" );
           return 0;
       }
   }
```

funkcia void PrintStack(Stack S):

```
void PrintStack( Stack S ){
            if (!IsEmpty(S)){
            TElement Temp = S->TopOfStack;
            while (Temp >= 0){
                printf("%d ",S->Array[Temp--]);
            }
            }else{
                Error("Stack is empty!");
            }
        }
```

funkcia main():

```
int main( )
{
    Stack S;
    int i;

    S = CreateStack( 12 );
    printf( "PUSH:\n");
    for( i = 0; i < 10; i++ )
    {
        printf( "%d ", i);
        Push( i, S );
    }
    putchar('\n');

    printf( "PrintList:\n");
    PrintStack(S);

    putchar('\n');

    printf( "TopAndPop:\n" );
    while( !IsEmpty(S))
    TopAndPop ( S );

    putchar('\n');

    printf("Is empty? (1=TRUE, 0=FALSE): %d\n", IsEmpty(S));

    DisposeStack( S );

    return 0;
}
```

Implementácia údajovej štruktúry zásobník s využitím zoznamu

Znenie zadania:

Pridajte podporu operácie Push pre vkladanie prvkov do zásobníka.
Prototyp operácie Push (stackl.h):

```
void Push( TElement X, Stack S );
```

Pridajte podporu operácie PrintStack, ktorá vypíše obsah zásobníka, bez jeho modifikácie.
Prototyp operácie PrintStack:

```
void PrintStack( Stack S );
```

Poznámka: Pre overenie činnosti operácie doplňte aj modul pre testovanie.

Opis riešenia:

Funkcia TopAndPop má pridať prvok na vrchol zásobníka. Najprv sa vytvorí pomocná premenná typu Stack, ktorá bude reprezentovať vytvorený prvok. Pre nový prvok sa alokuje miesto v pamäti o veľkosti štruktúry Node. Smerník na toto pamäťové miesto sa uloží do Temp. Do novo pridaného prvku sa uloží smerník na jeho nasledovníka – začiatok pôvodného stacku, a hodnotu novo vytvoreného prvku. Nakoniec sa smerník na tento novo vytvorený prvok uloží ako nasledovník nultého prvku v zásobníku.

Funkcia PrintStack má vypísať zásobník. Pomocná premenná typu Stack sa inicialuzuje na smerník na začiatok zásobníka – spraví sa tak záloha toho pointera. Pôvodný smerník sa potom nastaví na začiatok zoznamu. Nasleduje cyklus v ktorom sa postupne vypisujú hodnoty jednotlivých prvkov a posúva sa smerník na daný prvok. Cyklus sa opakuje až kým daný prvok nemá nasledovníka. Tento prvok je tým pádom posledný, čiže sa vypíše a ako smerník na začiatok zoznamu sa opäť nastaví pôvodná hodnota zálohovaná v Temp.

Pre otestovanie funkcií som vo funkcii main cyklus kde sa postupne volá funkcia push, aby naplnila zásobník číslami od 0 po 10. Tento zásobník sa vypíše pomocou funkcie PrintStack. Nakoniec overujem či funkcia PrintStack neovplyvnila zásobník jej duplicitným volaním.

Zdrojový kód:

funkcia void Push (TElement X, Stack S):

```
void Push( TElement X, Stack S){
      Stack Temp;
      Temp = malloc( sizeof( struct Node ) );
      Temp->Next = S->Next;
      Temp->Element = X;
      S->Next = Temp;
   }
```

funkcia void PrintStack(Stack S):

```
void PrintStack( Stack S ){
            if (!IsEmpty(S)){
            Stack Temp = S;
            S = S->Next;
            while (S->Next != NULL){
                printf("%d ",S->Element);
                S = S->Next;
            }
            printf("%d ",S->Element);
            S = Temp;
            }else{
                Error("Stack is empty!");
```

```
                }
            }
```

funkcia main():

```
int main( )
{
    Stack S;
    int i;

    S = CreateStack( );

    if(IsEmpty( S )) printf("Empty stack");
    else printf("Non-empty stack");

    putchar('\n');

    for (i=0; i<10; i++) Push(i, S);

    printf("PrintStack (#1): ");
    PrintStack(S);

    putchar('\n');

    printf("PrintStack (#2): ");
    PrintStack(S);

    putchar('\n');

    DisposeStack( S );

    return 0;
}
```

Implementácia údajovej štruktúry front (pomocou poľa)

Znenie zadania:

Pridajte podporu operácií Front, Dequeue, FrontAndDequeue.

Prototypy operácií:

```
TElement Front( Queue Q );
void Dequeue( Queue Q );
TElement FrontAndDequeue( Queue Q );
```

Pridajte podporu operácie PrintQueue, ktorá vypíše obsah frontu, bez jeho modifikácie.

Prototyp operácie PrintQueue:

```
void PrintQueue( Queue Q );
```

Poznámka: Pre overenie činnosti operácie doplňte aj modul pre testovanie.

Opis riešenia:

Funkcia Front po overení či je front prázdny, vráti prvok zo začiatku daného frontu.

Funkcia Dequeue má odobrať prvok zo začiatku frontu. To sa deje zvýšením indexu na prvý prvok a zároveň znížením veľkosti frontu o jedna.

Funkcia FrontAndDequeue má vrátiť prvý prvok a následne ho z frontu odstrániť. Najprv sa do pomocnej premennej uloží prvý prvok. Potom sa odstráni tak isto ako vo funkcii Dequeue. Nakoniec funkcia vráti prvok ktorý je uložený v premennej Temp.

Funkcia PrintQueue má vypísať front. Opäť sa najprv vytvorí pomocná premenná, ktorá bude odkazovať na prvok ktorý sa práve vypisuje. Začína sa vypísaním prvého prvku čiže Temp sa inicialuzuje na hodnotu adresy poľa kde je prvý prvok. Nasleduje cyklus v ktorom sa vypisuje prvok poľa s indexom ktorý je uložený v Temp a Temp sa inkrementuje. Tento cyklus sa opakuje až po posledný prvok, čiže kým sa Prvok poľa nerovná NULL.

Aby som otestoval vytvorené funkcie som ich postupne volal vo funkcii main. Funkciu PrintQueue som opäť zavolal duplicitne aby som overil, že funkcia neovplyvňue front.

Zdrojový kód:

funkcia TElement Front(Queue Q):

```
TElement Front(Queue Q){
    if(!IsEmpty(Q)){
    return Q->Array[Q->Front];
    }else{
        Error("Queue is empty");
    }
}
```

funkcia void Dequeue(Queue Q):

```
void Dequeue( Queue Q){
         if(!IsEmpty(Q)){
             Q->Front ++;
             Q->Size--;
         }else{
             Error("Queue is empty");
         }
     }
```

funkcia TElement FrontAndDequeue(Queue Q):

```
TElement FrontAndDequeue( Queue Q ){
      if(!IsEmpty(Q)){
          int Temp = Q->Array[Q->Front];
          Q->Front ++;
          Q->Size--;
          return Temp;
      }else{
          Error("Queue is empty");
      }
  }
```

funkcia void PrintQueue(Queue Q):

```
void PrintQueue(Queue Q){
      if (!IsEmpty(Q)){
      TElement Temp = Q->Front;
      while (Q->Array[Temp] != NULL){
          printf("%d ",Q->Array[Temp++]);
      }
      }else{
          Error("Stack is empty!");
      }
  }
```

funkcia main():

```
int main( )
{
    Queue Q;
    int i;

    Q = CreateQueue( 12 );

    if(IsEmpty( Q )) printf("\nEmpty queue.");
    else printf("\nNon-empty queue.");

    printf("\nEnqueue:\n");
    for( i = 0; i < 10; i++ ){
         printf( "%d ", i );
         Enqueue( i, Q );
    }

    if(IsEmpty( Q )) printf("\nEmpty queue.");
    else printf("\nNon-empty queue.");

    printf("\nFront: %d", Front(Q));
    Dequeue(Q);
    printf("\nDequeue: %d", Front(Q));
    printf("\nFrontAndDequeue: %d", FrontAndDequeue(Q));

    printf("\nPrintQueue (#1): ");
    PrintQueue(Q);
    printf("\nPrintQueue (#2): ");
    PrintQueue(Q);

    DisposeQueue( Q );
    return 0;
}
```