Guía de Practica N° 2 Pilas

Código de referencia: eedd_dinamicas_liniales_1.cpp

Referencias:

LSE Pilas

Recomendaciones:

- ☐ Utilizar funciones para el desarrollo.
- ☐ Probar los programas, en la máquina o con pruebas de escritorio.
- ☐ Utilizar nombres de fácil lectura para las variables.

```
struct nodo_pila
{  int dato;
    struct nodo_pila* link;
};
typedef struct nodo_pila NPila;
```

1. Dada una PILA de nombre S del tipo NPila, vacía, cargue los valores 5,3 y 8. Luego obtenga un valor de S y muestre por consola.

```
Luego mostrar el estado de la pila S
vacia={verdadero, falso}
```

2. Dada una PILA de nombre S del tipo NPila, vacía, realice una función que la cargue con los valores incluidos en un vector de enteros pasado como argumento. A modo de ejemplo se pide probar esta función pasando como parámetro el vector int v[] {9,1,3};

Luego implemente otra función que descargue completamente la pila mostrando su contenido por consola:

```
Retiramos: 3, 1, 9

Luego mostrar el estado de la pila S

vacia={verdadero, falso}
```

- 3. Implemente una función que invierta una palabra usando una pila tipo SPila. Ej entrada string p = "acamah"; Salida string in = "hamaca" Recordar que un char tiene una representación interna de un entero de 0 a 255.
- **4.** Se desea realizar una función de nombre dec2bin que reciba como parámetro un número en formato de numeración decimal y obtenga la representación binaria del mismo. Implemente la solución usando una pila del tipo NPIla.

Para implementar esta función recordamos el proceso de conversión:

```
75_{(10} = 1001011_{(2)}
```

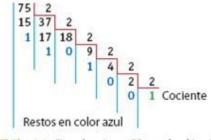


Fig. 1.1. Ejemplo: número 75 pasado a binario.

- 5. Implemente una solución alternativa para el problema del punto anterior de nombre dec2binrec, usando una función recursiva solamente (sin estructuras auxiliares). Complementariamente debe representar en papel el estado de invocación y del stack, con el caso del decimal 7.
- **6.** Genere una función de nombre capicua, que utilice una pila del tipo NPila, que reciba como entrada un numero entero (int) y determine si el mismo es capicúa.
- 7. Implemente una función llamada calculadorapolaca, que permita resolver una expresión algebraica con constantes enteras de un digito. Utilice para su desarrollo el siguiente prototipo

```
float calculadorapolaca(string expresión) ;
```

Ej entrada string exppolaca = "512+4*+3-"; Salida in sal = 14 Exppolaca corresponde con la expresión algebraica: y=5+(1+2)*4-3

Para resolución deberá consultar:

https://es.wikipedia.org/wiki/Notaci%C3%B3n_polaca_inversa

8. Dado el siguiente archivo de cabecera (header) complete la codificación de las funciones PilaE_vacia, PilaE_agregar y PilaE_obtener, correspondientes a la implementación de una pila estática de enteros.

```
PilaE int.h
  #define MAX 20
                ******
  // Implementación de una Pila Estática de enteros
  // UADER - Algoritmos y Estructuras de Datos
  class PilaLLena : public exception{}
  class PilaVacia : public exception{};
  struct pila_estatica
  {
    int dato[MAX];
    int tamanio;
    int tope;
  typedef struct pila estatica PilaE;
  void PilaE inicializa (PilaE &pila, int maximo) {
  }
  bool PilaE vacia(PilaE pila) {
  }
  void PilaE agregar(PilaE &pila, int) {
  int PilaE obtener(PilaE &pila) {
```

Utilice el archivo de cabecera PilaE_int.h, para producir una nueva versión de dec2bin estint.

9. Dado el siguiente archivo de cabecera (header) complete la codificación de las funciones PilaG_vacia, PilaG_agregar y PilaG_obtener, correspondientes a la implementación de una pila estática genérica que permita alojar objetos primitivos (char, int, float, string) u objetos definidos por el usuario.

```
PilaEg generica.h
  #define MAX 20
                  ************************************
  // Implementación de una \, Pila Estática de (void*), que permita
      alojar cualquier tipo de elementos, elementos primitivos:
  // char, int, float, string, e incluso estructuras y objetos
  // definidos por el usuario Personas, objetos de distinto tipo.
  // UADER - Algoritmos y Estructuras de Datos
  #define MAX 20
  using namespace std;
  class PilaLLena : public exception{};
  class PilaVacia : public exception{};
  struct pila_estatica
  { void * pelemento[MAX];
     int tamanio;
     int tope;
  typedef struct pila estatica PilaEg;
  void PilaEg inicializa (PilaEg &pila, int maximo) {
  bool PilaEg vacia (PilaEg pila) {
  void PilaEg agregar (PilaEg &pila, void * pelemento) {
  void * PilaEg obtener(PilaEg &pila) {
  Para uso con primitivo int:
main()
  PilaEq pila;
  int * pdato;
  // para hacer push
  pdato = new(int);
  *pdato = ;
  PilaEg agregar(pila, (void *) pdato);
  // para pop
  pdato = (int *) PilaEg obtener(pila);
  Para uso con TAD:
  struct spersona {
     int dni;
     char nombre[40];
  };
main() {
     PilaEg pila;
     spersona *pdato;
     // para hacer push
     pdato = new(spersona);
     (*pdato).dni
                     = 13241227;
     strcpy((*pdato).nombre, "Maradona Diego");
     PilaEg_agregar(pila, (void *) pdato);
     // para hacer pop
     pdato = (spersona *) PilaEg obtener(pila);
```