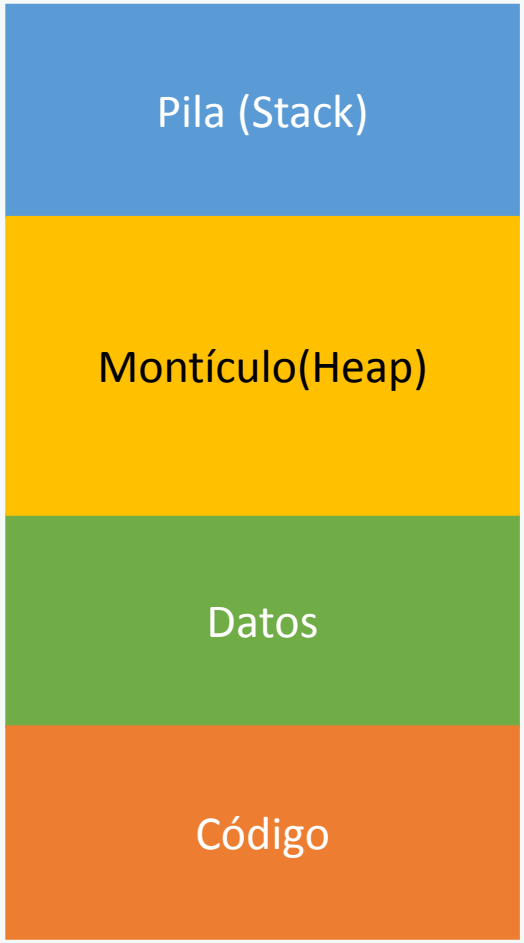
CURSOS INTERSEMESTRALES

PROTECO

C++ intermedio

Apuntadores

Estructura de un programa en memoria



En la pila se localizan datos que se utilizan con las funciones.

En el montículo se encuentran datos asignados dinámicamente así como datos de bibliotecas compartidas

En el área de datos tenemos las variables y constantes definidas

En el área de código están las instrucciones del programa.

PROTECO

Direcciones y referencias

Dirección: Con esta palabra nosotros indicaremos las localidades de memoria en nuestra computadora

Referencia: Haremos uso de esta palabra al hacer mención de "etiquetas" (nombres de variable) con las que identificaremos las direcciones

Contenido: Como su nombre lo indica, el contenido de las localidades de memoria.

PROTECO

Direcciones y referencias

Direcciones	Referencia	Contenido
700	x	100
600		
500	y	???
400		
300		
200	Z	3
100		
0		

PROTECO

Apuntadores

En C++, nosotros podemos conocer las direcciones de memoria y contenido de éstas gracias a los dos siguientes operadores

Operador de referencia (*): Nos devuelve el contenido de la localidad referenciada por la variable. Se usa únicamente para apuntadores.

Operador de desreferencia(&): Nos devuelve la dirección en memoria de la variable junto a este operador. Se utiliza para variables y apuntadores

PROTECO

Apuntadores

Para declarar una variable de tipo apuntador.

```
<tipo_dato>* <nombre_variable> = <dirección>;
```

Ejemplos

```
//Creamos un apuntador a la variable x.
int* ap_int = &x;
//Declaramos un apuntador nulo (apunta a nada)
char* ap_char = NULL;
//Después asignamos a dónde apunta
ap_char = &letra;
```

PROTECO

Apuntadores

Ejemplos

Direcciones	Referencia	Contenido
700	int x	450
600		
500	int y	???
400		
300		
200		
100	int* z	700
0		

Al querer saber el valor:

x nos devuelve 450
&x nos devuelve 700

y nos devuelve algo (no sabemos)
&y nos devuelve 500

z nos devuelve 700
*z nos devuelve 450
&z nos devuelve 100

PROTECO

Funciones con apuntadores como parámetros

Pasar apuntadores como parámetros nos permite una nueva forma de pasar parámetros por referencia a una función.

```
void incrementar(int* x) {
     (*x)++;
}...
int x = 10;
incrementar(&x);
int* ap_x = &x;
Incrementar(ap_x);
```

Los operadores * y & tiene una precedencia menor a los operadores aritméticos, lógicos, relacionales y condicionales.

PROTECO

Funciones con apuntadores como parámetros

También se pueden utilizar apuntadores para pasar arreglos o matrices como parámetros a una función.

```
void llenarArreglo(int* arreglo,int tamano);


//Matrices dinámicas (las veremos más adelante)
void leerMatriz(int** matriz,int col,int ren);
```

PROTECO

Arreglos de apuntadores

Nos ayudan a manejar matrices con apuntadores.

Cada apuntador del arreglo apunta a un renglón de la matriz, y cada renglón lo manejamos como un arreglo común. No hay que olvidar inicializar cada apuntador del arreglo.

```
int matriz[renglones][columnas];
int  *ap_matriz[columnas];
for(int i = 0; i < renglones; i++) {
     ap_matriz[i] = matriz[i];
}
```

Manejar una matriz de este modo puede ser un poco tedioso, se recomienda utilizar la forma vista en el tema de Funciones.

Apuntador a estructuras

```
struct <nombre_struct>* <ap> = <dirección>;
```

Ejemplos:

```
struct alumno ap_alumno = &alumno_1;
```

Para acceder a elementos de estructuras mediante apuntadores utilizamos el operador flecha (->).

```
cout << alumno_1.nombre;
cout << ap_alumno->nombre;
```

PROTECO

Apuntador a apuntadores

Al ser los apuntadores variables, también podemos crear un apuntador que apunte a ellos.

```
//Declaramos una variable x
int x;
//Declaramos un apuntador a esa variable x
int* ap_x = &x;
//Declaramos un apuntador a dicho apuntador
int** ap_ap = &ap_x;
//Modificando x con el apuntador a apuntador
**ap_ap = 10;
```

PROTECO

Apuntador a apuntadores

Útiles para el manejo de matrices dinámicas y demás arreglos multidimensionales.

Esto se debe a que con un apuntador podemos "navegar" en un arreglo. Análogamente un apuntador a apuntador nos permite "navegar" en ese arreglo de apuntadores.

PROTECO

Apuntador a funciones

Las funciones tienen un lugar en la memoria, por lo tanto también podemos definir apuntadores hacia ellas.

```
<tipo_dato> (*<nombre_ap>)(<tipo_parametros>);
```

```
//Apuntador a función que devuelve un entero y que tiene un parámetro de tipo entero.
int (*ap_funcion)(int);
//Apuntador a función que no devuelve datos y que tiene un parámetro de tipo apuntador a entero y uno de tipo carácter.
void (*ap_funcion)(int*,char);
```

PROTECO

Apuntador a funciones

Estos apuntadores son útiles para casos en que usemos bibliotecas cuyas funciones acepten como parámetros funciones hechas por el usuario (como en algunas bibliotecas de C) y aceptan un apuntador a función.

PROTECO