Clase #21 de 29

Ejemplo de Especificación & Clases de Almacenamiento

Sep 23, Martes
Sep 24, Miércoles

Sintaxis & Semántica de for y while

For versus While

```
// Una sentencia for equivalente a una sentencia while

while ( expresión ) sentencia

for ( ; expresión ; ) sentencia


// "Una" sentencia while equivalente a una for

for ( expresión1; expresión2 ; expresión3) sentencia

expresión1;
while ( expresión2 ) { sentencia expresión3; }
```

Sintaxis y Semántica Simple de While y de For

- While
  - Sintaxis
    - *SentenciaWhile* → while (*Expresión*) *Sentencia*
    - *Sentencia* → ...
    - *Expresión* → ...
  - Semántica
    - while (*Expresión1*) *Sentencia1*
      - LN
- For (C90)
  - Sintaxis
    - *SentenciaFor* → for ($Expresión_{opt}$; $Expresión_{opt}$; $Expresión_{opt}$)*Sentencia*
  - Semántica
    - for ($Expresión1_{opt}$; $Expresión2_{opt}$; $Expresión3_{opt}$) *Sentencia1*
      - LN.

Sintaxis y Semántica Simple de While y de For

- For (C90)
  - Sintaxis
    - *SentenciaFor* → for ($Expresión_{opt}$; $Expresión_{opt}$; $Expresión_{opt}$) *Sentencia*
  - Semántica
    - for ($Expresión1_{opt}$; $Expresión2_{opt}$; $Expresión3_{opt}$) *Sentencia1*
      - LN
- For (C99)
  - Sintaxis
    - *SentenciaFor* → for ($Expresión_{opt}$; $Expresión_{opt}$; $Expresión_{opt}$) *Sentencia*
    - *SentenciaFor* → for (*Declaración1* $Expresión_{opt}$; $Expresión_{opt}$) *Sentencia*
  - Semántica
    - for ($Expresión1_{opt}$; $Expresión2_{opt}$; $Expresión3_{opt}$) *Sentencia1*
      - LN.
    - for (*Declaración1* $Expresión1_{opt}$; $Expresión2_{opt}$) *Sentencia1*
      - LN.

Ejercicio de Investigación

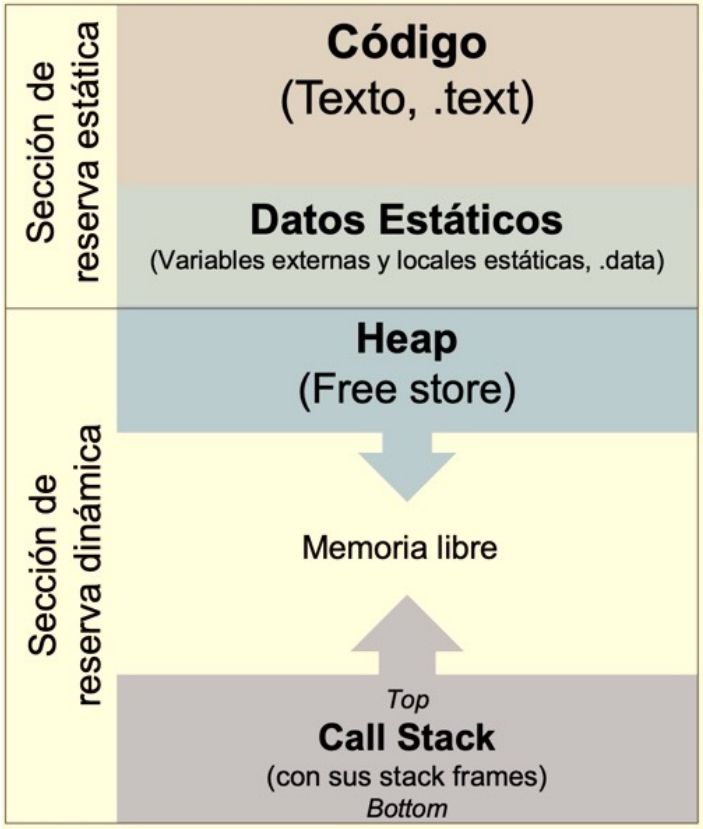
- S&S de:
  - for(;;)
  - C++ for(e:a)s

Clases de Almacenamiento

Organización de Memoria

Repaso de AED

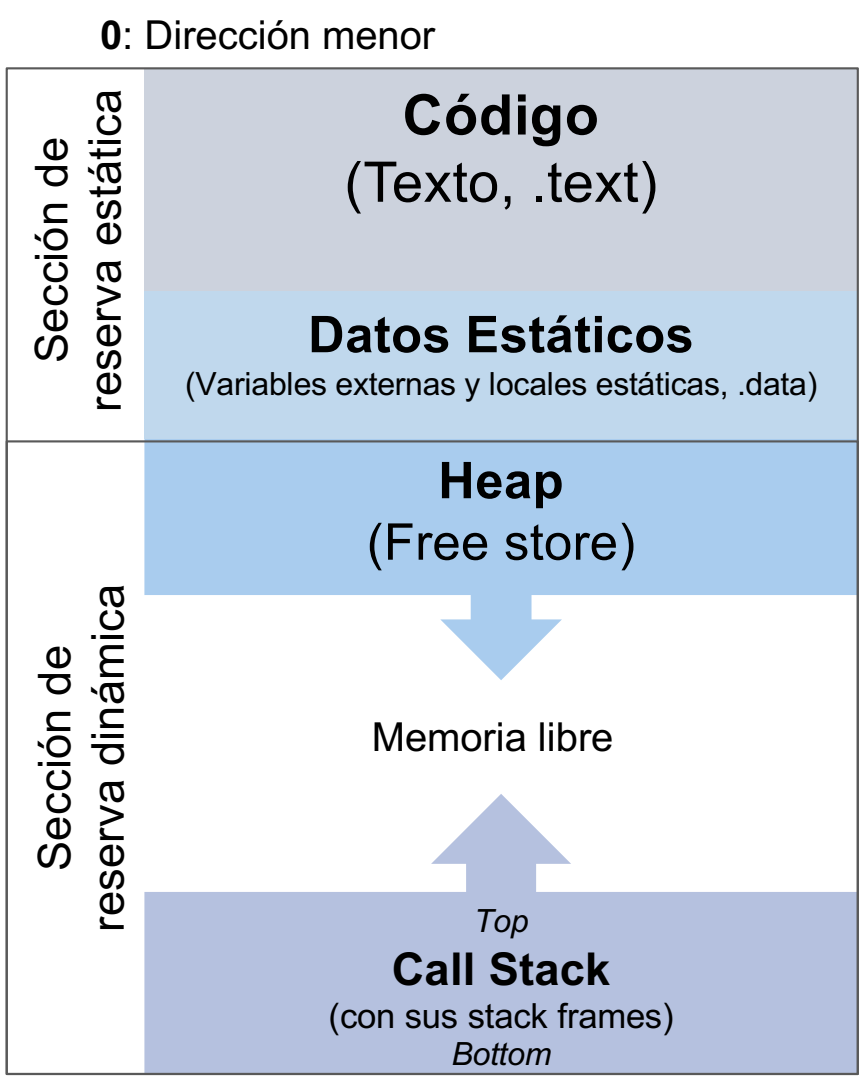
Layout (Disposición) de la Memoria




```
void f();
int* g(int);

static int a;        // Static

int main(){
  int b{1};          // Stack (automatic)

  f();

  int* p{&b};        // Stack
  *p = 3;

  p = new int;       // Heap
  *p = 4;
  delete p;

  p = g(21);
  delete p;
}


void f(){
  int c{2};          // Stack
  extern int a;      // No definition,just declaration
  ++a;
  static int d{a}; // Static
  ++d;
}

int* g(int i){       // Stack
  int* p{new int}; // Stack y Heap
  *p = i;
  return p;
}
```

Esp. Ing. José María Sola, Profesor 376

Memory Layout, Storage Duration, Lifetime, Scope, Linkage




```c
#include <stdlib.h>// malloc exit free

void f();
int *g(int);

static int a;       // Static

int main(){
   int b=1;         // Stack (automatic)

   f();

   int *p = &b;     // Stack
   *p = 3;

   p = malloc( sizeof *p );// Heap (allocated)
   if(NULL==p) exit(1);    // Not enough memory
   *p = 4;
   free(p);

   p = g(21);
   free(p);
}


void f(){
   int c = 2;       // Stack
   extern int a;    // No definition, just declaratio
   ++a;
   static int d=7; // Static
   ++d;
}

int *g(int i){                      // Stack
   int *p = malloc( sizeof *p );// Stack & Heap
   if(NULL==p) exit(1);             // Not enough memory
   *p = i;
   return p;
}

//
```

C++ vs C – Tipos de Duraciones del Almacenamiento de Objetos Atuomática, Estática, *Alocada* (y Local a Thread)

```
void f();
int* g(int);

static int a;       // Static

int main(){
    int b{1};       // Stack (automatic)

    f();

    int* p{&b};     // Stack
    *p = 3;

    p = new int;    // Heap
    *p = 4;
    delete p;


    p = g(21);
    delete p;
}

void f(){
    int c{2};       // Stack
    extern int a;   // Declaration,not definition
    ++a;
    static int d{a}; // Static
    ++d;
}

int* g(int i){      // Stack
    int* p{new int}; // Stack & Heap
    *p = i;
    return p;
}
```

```
#include <stdlib.h>                 // malloc exit free

void f();
int *g(int);

static int a;                       // Static

int main(){
    int b = 1;                      // Stack (automatic)

    f();

    int *p = &b;                    // Stack
    *p = 3;

    p = malloc( sizeof *p );        // Heap
    if(NULL==p) exit(1);            // Not enough memory
    *p = 4;
    free(p);

    p = g(21);
    free(p);
}

void f(void){
    int c = 2;                      // Stack
    extern int a;                   // Declaration,not definitio
    ++a;
    static int d = 0;               // Static
    ++d;
}

int *g(int i){                      // Stack
    int *p = malloc( sizeof *p );   // Stack & Heap
    if(NULL==p) exit(1);            // Not enough memory
    *p = i;
    return p;
}
```

¿Consultas?

Fin de la clase