Apuntadores

Programación en C++

Topicos

- Apuntadores
  - Dirección de Memoria
  - Declaracion
  - Desreferenciar un apuntador
  - Apuntador a apuntador

Memoria del Computador

- Cada variable es asignada a una celda de memoria (el tamaño depende del tipo de dato) y el dato de la variable se almacena alli.

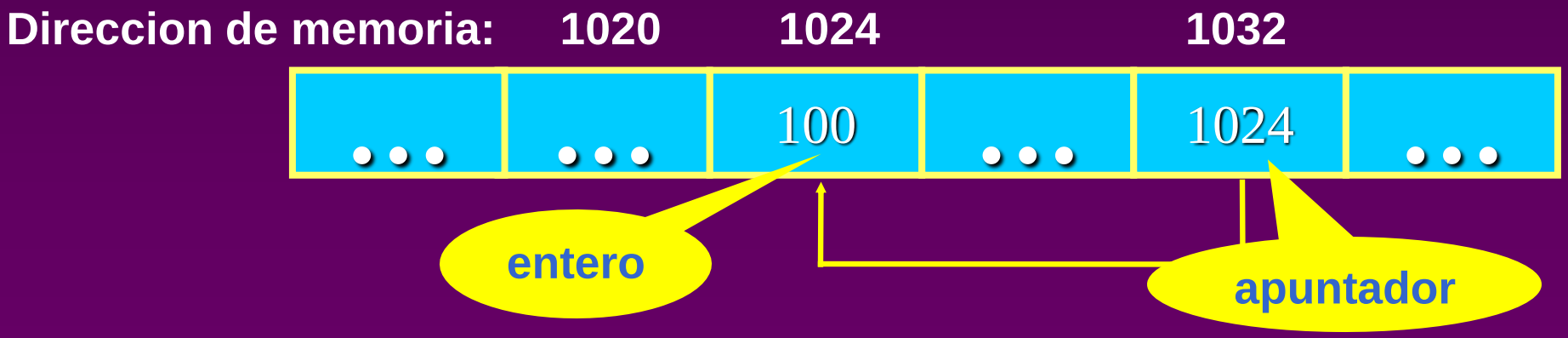
Dirección de Memoria:


```
int a = 100;
```

El valor de la variable a, i.e., 100, es almacenado en la dirección de memoria 1024

Apuntadores

- Un apuntador es una variable usada para almacenar la dirección de una celda de memoria.
- Podemos usar el apuntador para referenciar esta celda de memoria



Tipos de Apuntadores

- Apuntador
  - C++ tiene tipos de apuntadores para cada tipo de objeto
    - Apuntadores a objetos int
    - Apuntadores a objetos `char`
    - Apuntadores a objetos definidos (ejm, `NumeroRacional`)
  - Incluso apuntadores a apuntadores
    - Apuntadores a apuntadores a objetos enteros

Variable Apuntador

- Declaración de variables Apuntador

```
tipo* nombre_apuntador;
//or
tipo *nombre_apuntador;
```

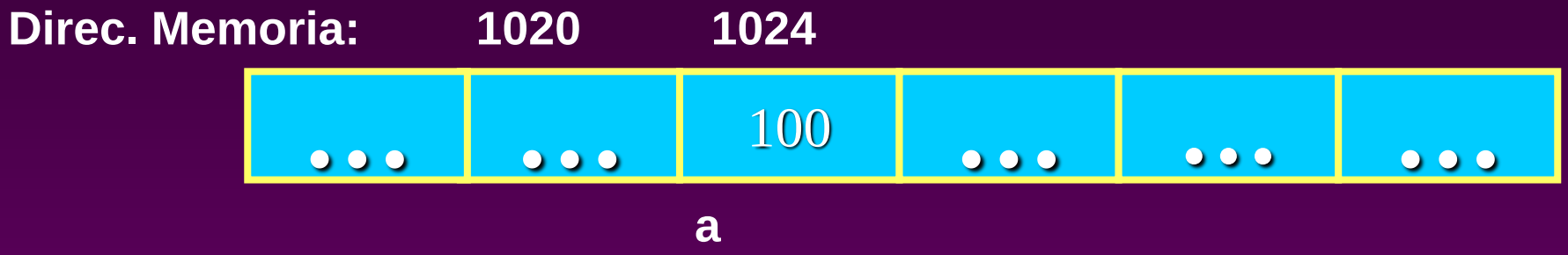
donde *tipo* es el tipo de dato a apuntar (ejm: int, char, double)

Ejemplos:

```
int *n;
NumeroRacional *r;
int **p;    // apuntador a apuntador
```

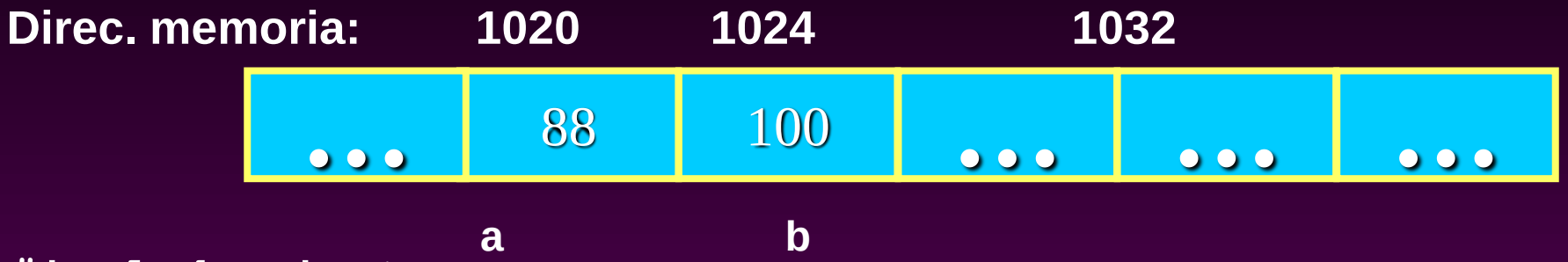
Operador de Dirección &

- *La* "dirección del " *operador* (&) da la dirección de memoria de la variable
  - **Uso:** `&nombre_variable`




```
int a = 100;
//obtiene el valor,
cout << a;    //imprime 100
//obtiene la direccion de memoria
cout << &a;    //imprime 1024
```

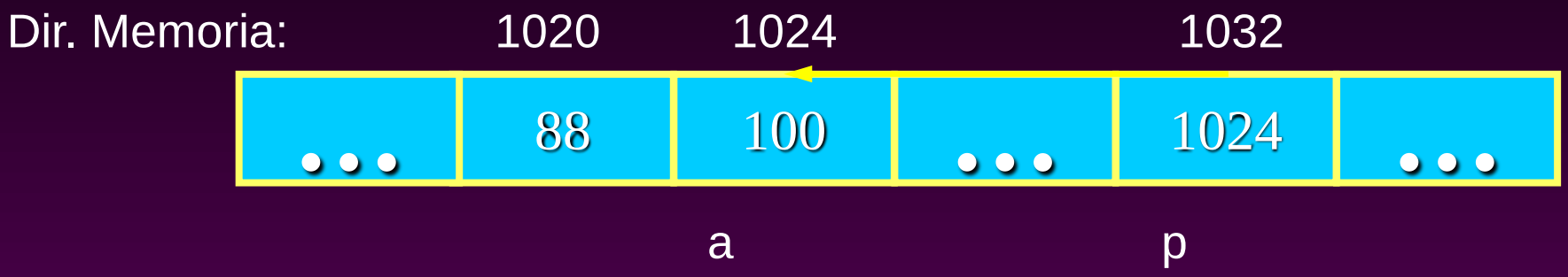
Operador de Dirección &




```
#include <iostream>

using namespace std;

void main(){

   int a, b;

   a = 88;

   b = 100;

   cout << "La direccion de a es: " << &a << endl;

   cout << "La direccion de b es: " << &b << endl;

}
```

- Resultado es:
  La direccion de a es: 1020
  La direccion de a es: 1024

Variables Apuntador




```
int a = 100;
int *p = &a;
cout << a << " " << &a <<endl;
cout << p << " " << &p <<endl;
```

- Resultado es:
  100 1024
  1024 1032

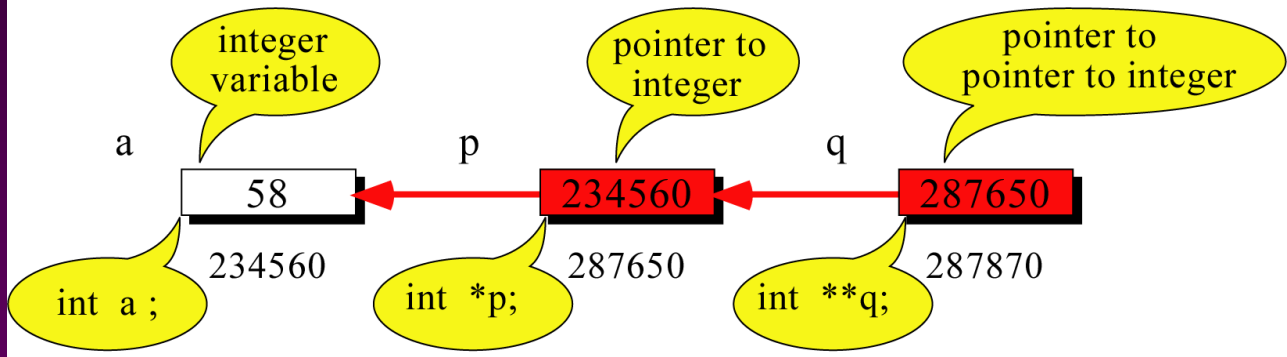
- El valor del apuntador `p` es la dirección de la variable `a`
- Un apuntador es también una variable, asi que esta tiene su propia direccion de memoria

Apuntador a Apuntador

¿Cuál es la salida?

58 58 58

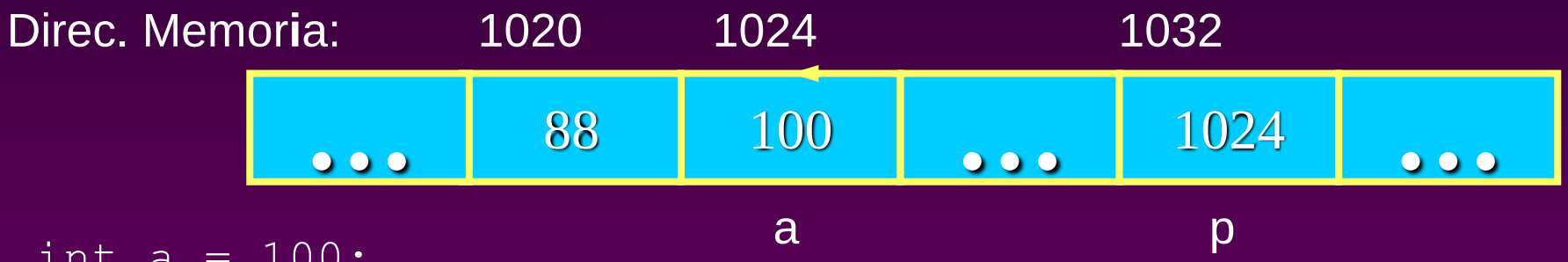
```
// Local Declarations
int         a ;
int         *p ;
int         **q ;
```




```
// Statements
a = 58 ;
p = &a ;
q = &p ;
cout <<       a << " ";
cout <<      *p << " ";
cout <<     **q << " ";
```

Operador de Indirección *

- Podemos acceder al valor almacenado en la variable apuntada usando el operador de indirección (*),




```
int a = 100;
int *p = &a;
cout << a << endl;
cout << &a << endl;
cout << p << " " << *p << endl;
cout << &p << endl;
```

- El resultado es:

100
1024
1024 100
1032

No confundir

- Declarar un apuntador significa que solo es un apuntador:

```
int *p;
```

- No confundir con el operador de indirección, el que también es escrito con un asterisco (*). Estas son simplemente dos tareas diferentes representadas por el mismo signo

```
int a = 100, b = 88, c = 8;
int *p1 = &a, *p2, *p3 = &c;
p2 = &b; // p2 apunta a b
p2 = p1;    // p2 apunta a a
b = *p3; //asigna c a la variable b
*p2 = *p3;  //asigna c a la variable a
cout << a << b << c;
```

El resultado es: 888

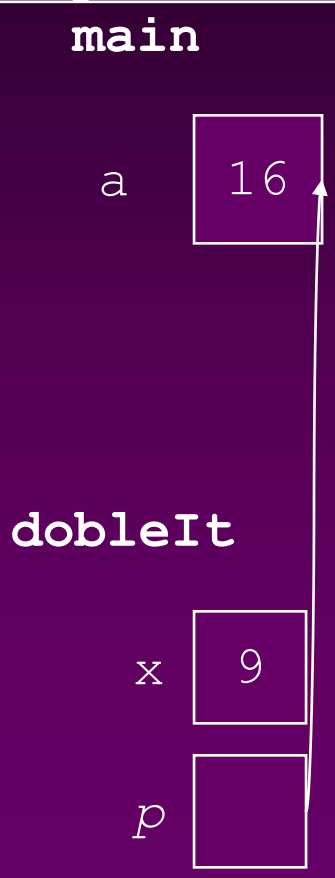
Un ejemplo de Apuntador

El código

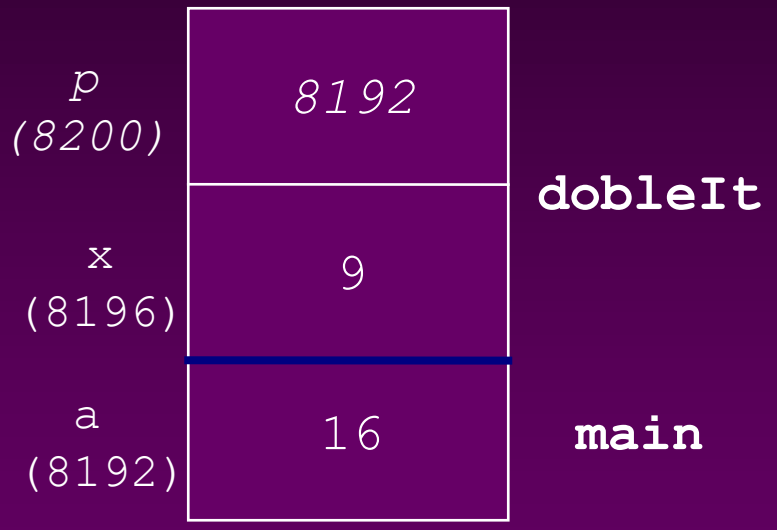
```
void dobleIt(int x,
             int * p)
{
   *p = 2 * x;
}
int main(int argc, const
   char * argv[])
{
   int a = 16;
   dobleIt(9, &a);
   return 0;
}
```

a obtiene 18

Diagrama de Caja



Esquema de Memoria



Otro Ejemplo de Apuntador

- **Averiguemos:**

valor1==? / valor2==?

También, p1=? p2=?

```
#include <iostream>
using namespace std;
int main (){
   int valor1 = 5, valor2 = 15;
   int *p1, *p2;
   p1 = &valor1; // p1 = direccion del valor1
   p2 = &valor2; // p2 = direccion del valor2
   *p1 = 10;     // valor apuntado por p1=10
   *p2 = *p1;    // valor apuntado por p2= valor
    // apuntado por p1
   p1 = p2;      // p1 = p2 (valor apuntador copiado)
   *p1 = 20;     // valor apuntado por p1 = 20
   cout << "valor1==" << valor1 << "valor2==" <<
   valor2;
   return 0;
}
```

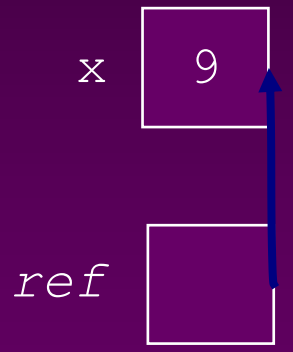
Otro Ejemplo de Apuntador

```
int a = 3;
char s  = 'z';
double d = 1.03;
int *pa = &a;
char *ps = &s;
double *pd = &d;
% sizeof retorna el numero de bytes…
cout << sizeof(pa) << sizeof(*pa)
     << sizeof(&pa) << endl;
cout << sizeof(ps) << sizeof(*ps)
     << sizeof(&ps) << endl;
cout << sizeof(pd) << sizeof(*pd)
     << sizeof(&pd) << endl;
```

Variable de Referencia

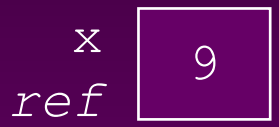
Una referencia es un nombre adicional a una ubicación de memoria existente

Apuntador:




```
int x=9;
int *ref;
ref = &x;
```

Referencia:




```
int x = 9;
int &ref = x;
```

Variable de Referencia

- Una variable de referencia sirve como un nombre alternativo para un objeto

```
int m = 10;
int &j = m;// j es una variable de referencia
cout << "valor de m = " << m << endl;
                    //  imprime 10
j = 18;
cout << "valor de m = " << m << endl;
                   //   imprime 18
```

Variable de Referencia

Una variable de referencia siempre refiere al mismo objeto. Asignar una variable de referencia con un nuevo valor cambia actualmente el valor del objeto referido.

- Las variables de Referencia son usadas frecuentemente para el paso de parámetros a funciones

Uso Tradicional de Apuntador

```
void SwapIndirecto(char *Ptr1, char *Ptr2){
   char temp = *Ptr1;
   *Ptr1 = *Ptr2;
   *Ptr2 = temp;
}
int main() {
   char a = 'y';
   char b = 'n';
   SwapIndirecto(&a, &b);
   cout << a << b << endl;
   return 0;
}
```

Paso por Referencia

```
void SwapIndirecto(char& y, char& z) {
   char temp = y;
   y = z;
   z = temp;
}
int main() {
   char a = 'y';
   char b = 'n';
   SwapIndirecto(a, b);
   cout << a << b << endl;
   return 0;
}
```