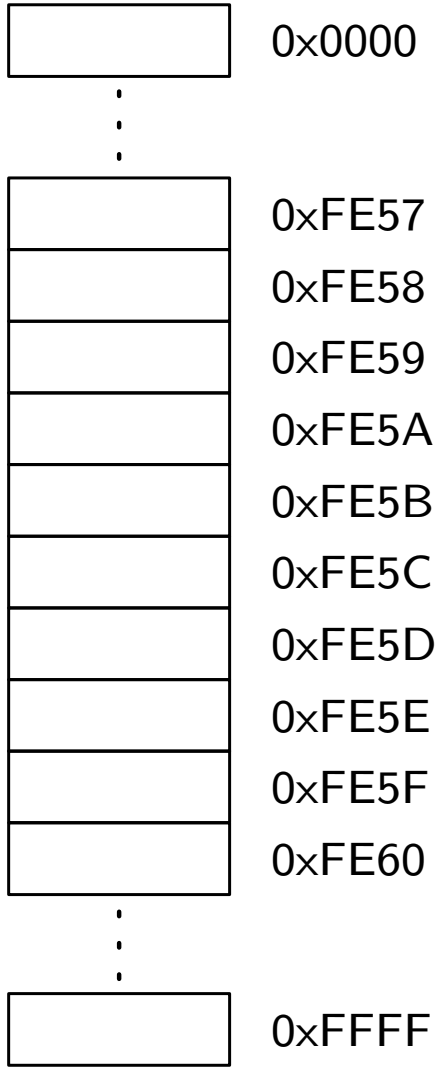
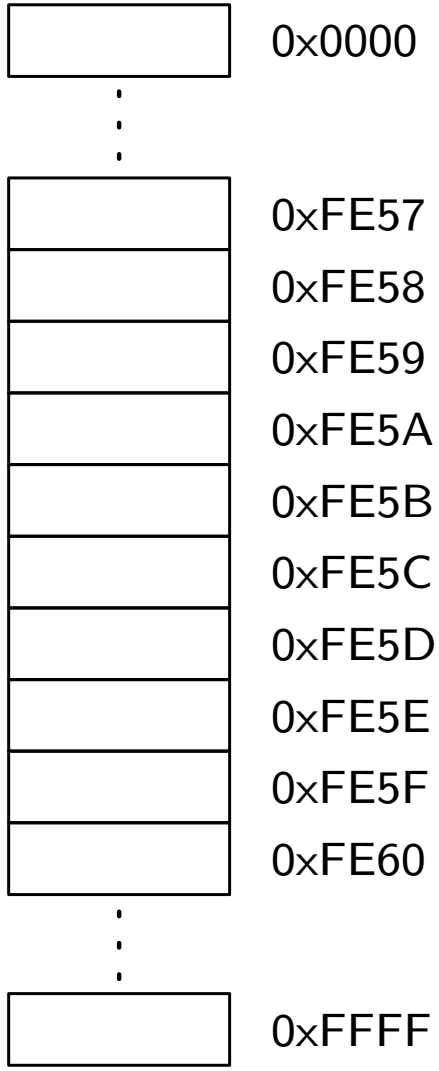
# Repaso


0x0000
0xFE57
0xFE58
0xFE59
0xFE5A
0xFE5B
0xFE5C
0xFE5D
0xFE5E
0xFE5F
0xFE60
0xFFFF

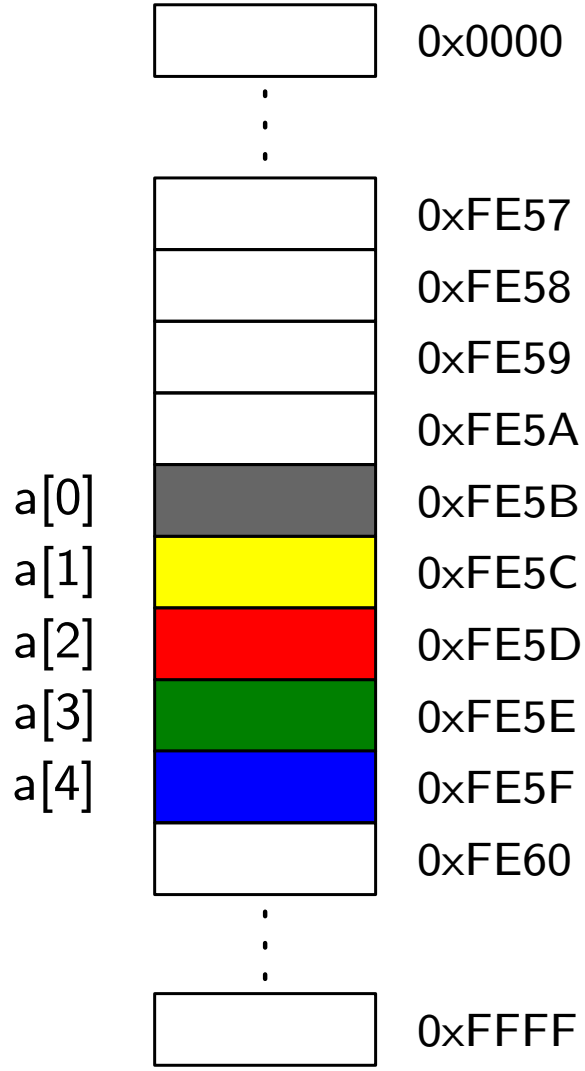
# Repaso

```
char a[5];
```

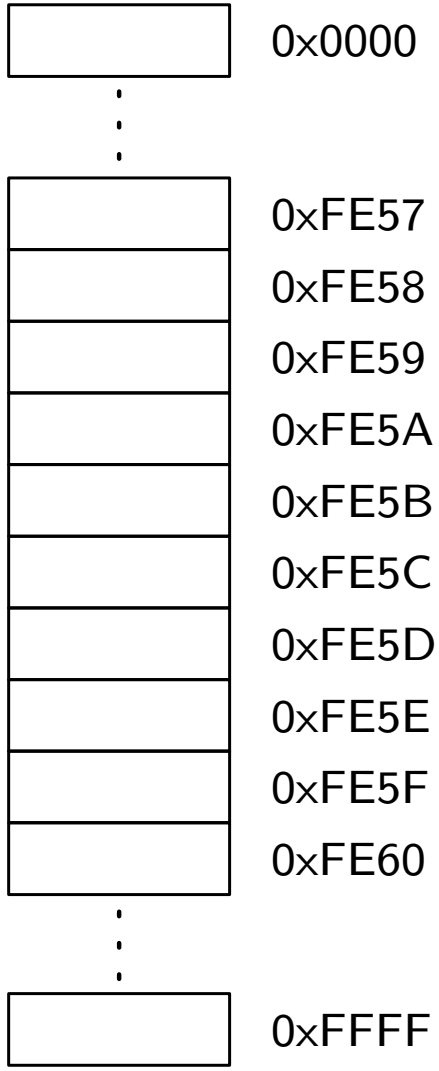


# Repaso

```
char a[5];
```

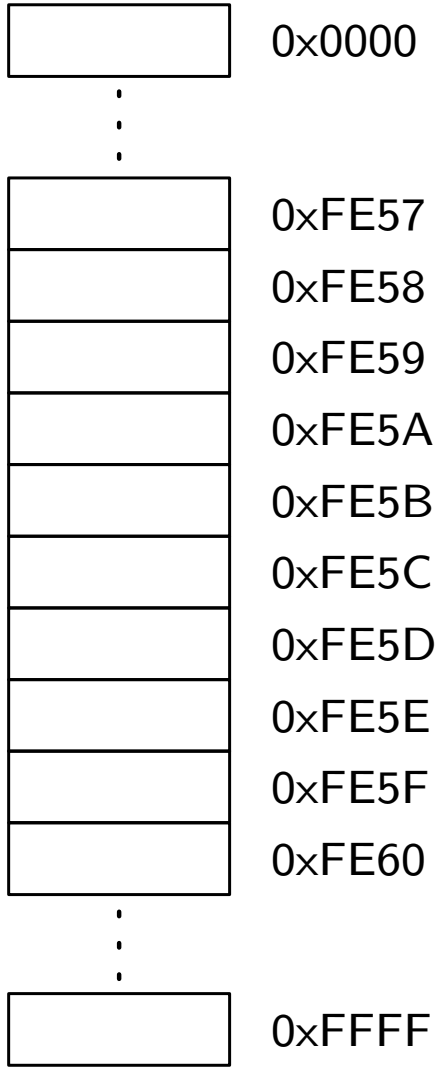


# Repaso


0x0000
0xFE57
0xFE58
0xFE59
0xFE5A
0xFE5B
0xFE5C
0xFE5D
0xFE5E
0xFE5F
0xFE60
0xFFFF

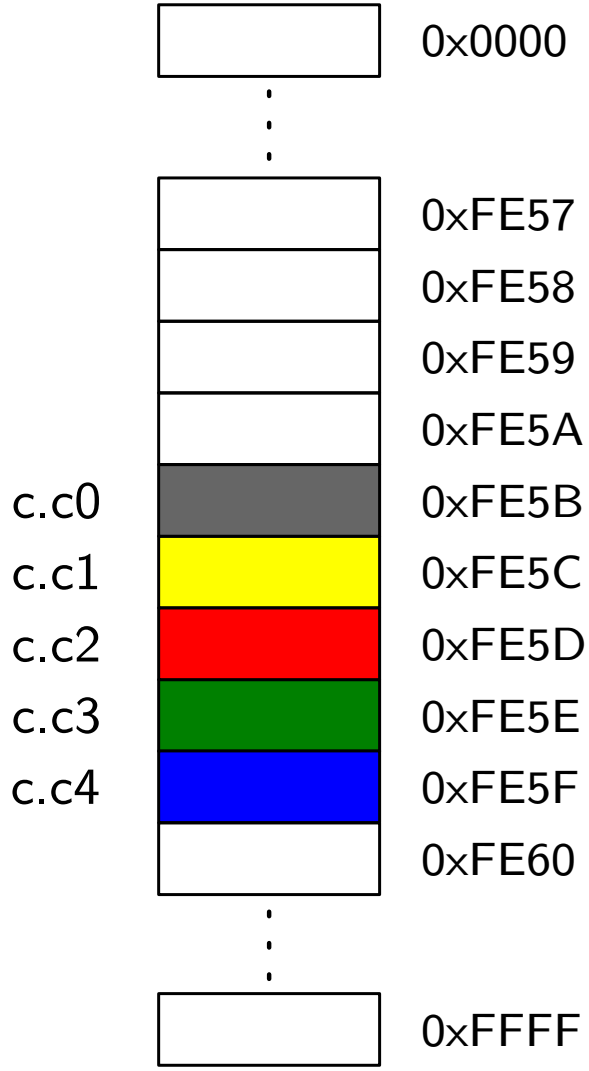
# Repaso

```
struct caracteres{
   char c0;
   char c1;
   char c2;
   char c3;
   char c4;
};

struct caracteres c;
```

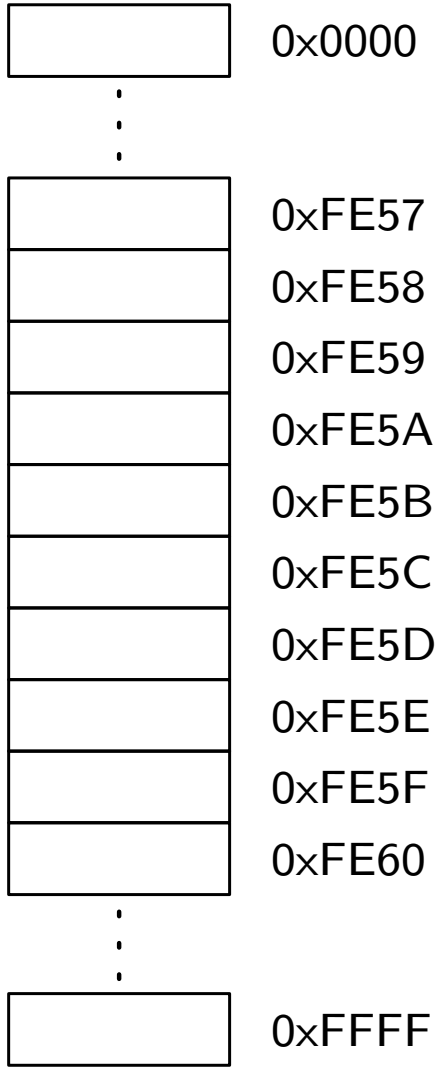


# Repaso

```
struct caracteres{
   char c0;
   char c1;
   char c2;
   char c3;
   char c4;
};

struct caracteres c;
```

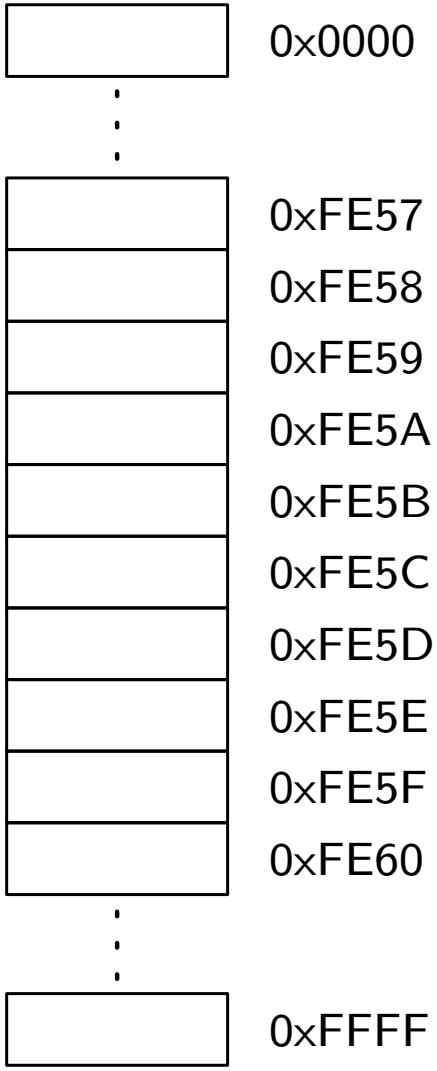


# Repaso


0x0000
0xFE57
0xFE58
0xFE59
0xFE5A
0xFE5B
0xFE5C
0xFE5D
0xFE5E
0xFE5F
0xFE60
0xFFFF

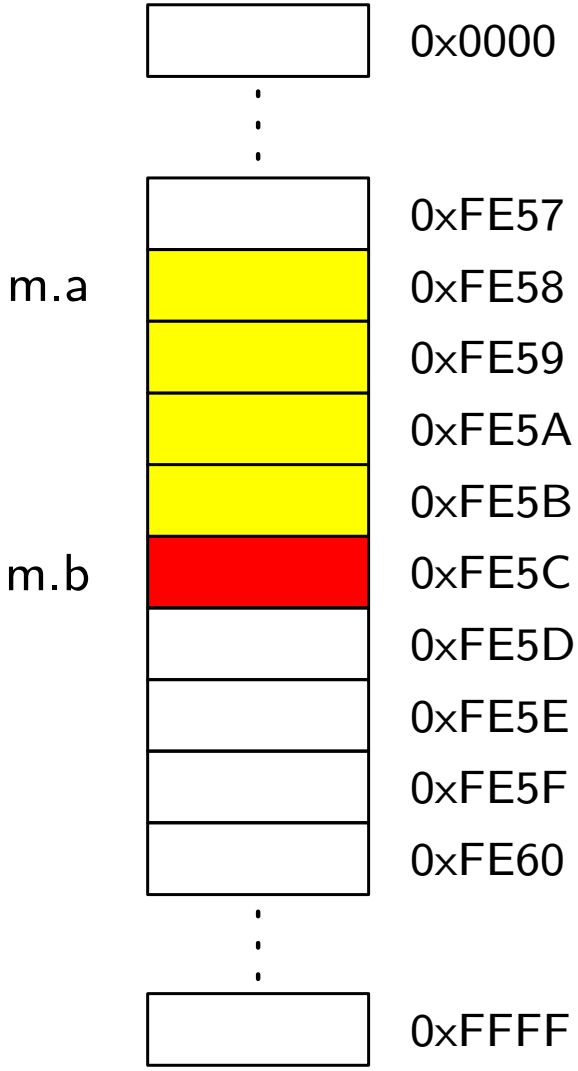
# Repaso

```
struct mixto {
   int a;
   char b;
};

struct mixto m;
```



# Repaso

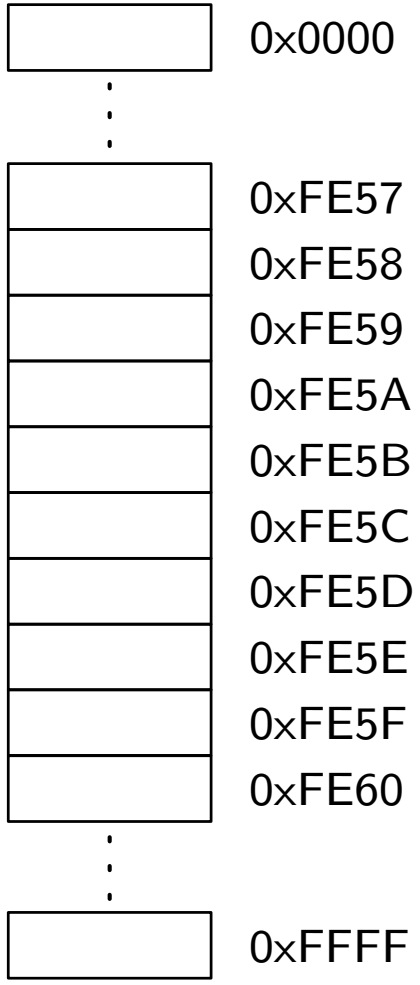
```
struct mixto {
   int a;
   char b;
};

struct mixto m;
```



# Uniones

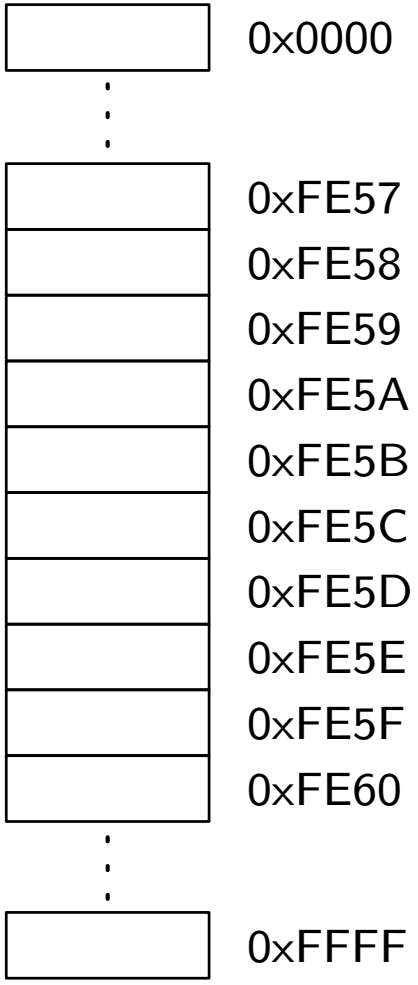
# Uniones
## Concepto

# Uniones
## Concepto



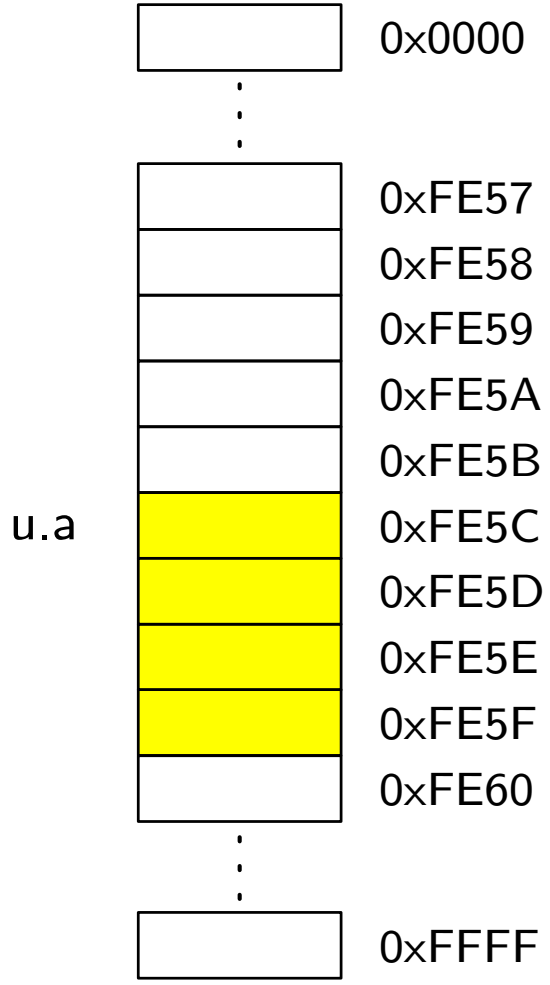
# Uniones
## Concepto

```
union int_float {
   int a;
   char b;
};

union int_float u;
```



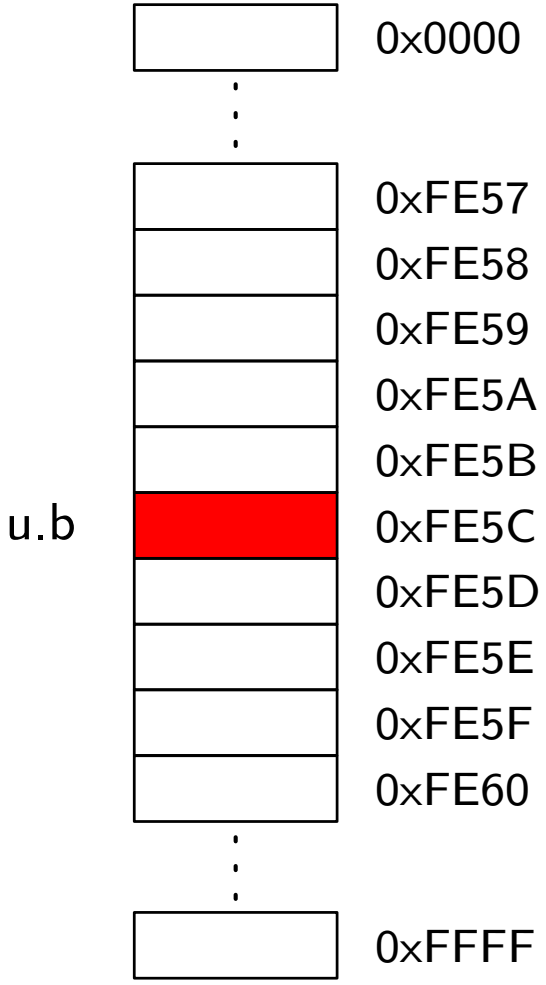
# Uniones
## Concepto

```
union int_float {
   int a;
   char b;
};

union int_float u;
```



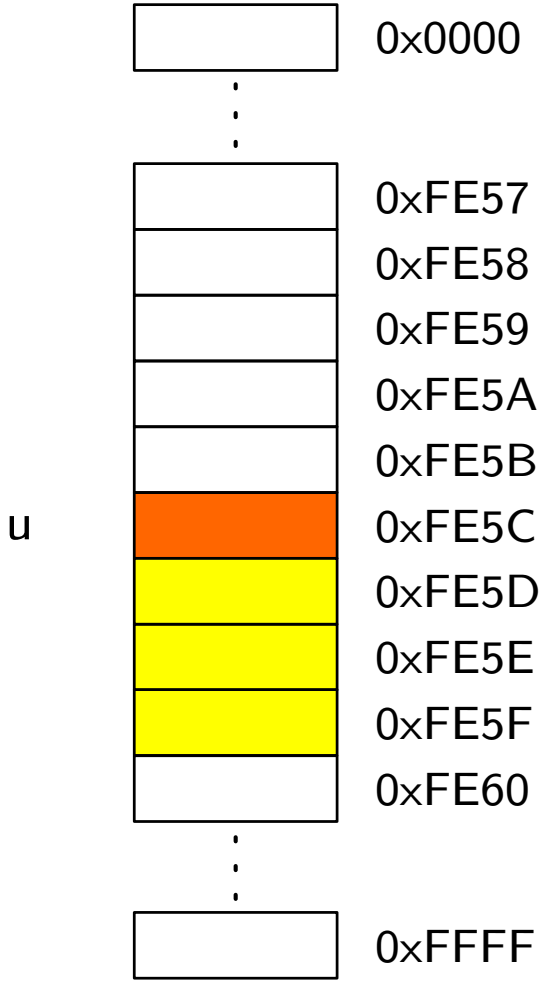
# Uniones
## Concepto

```
union int_float {
   int a;
   char b;
};

union int_float u;
```



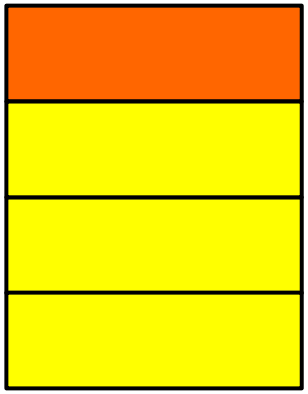
# Uniones
## Concepto

```
union int_float {
   int a;
   char b;
};

union int_float u;
```

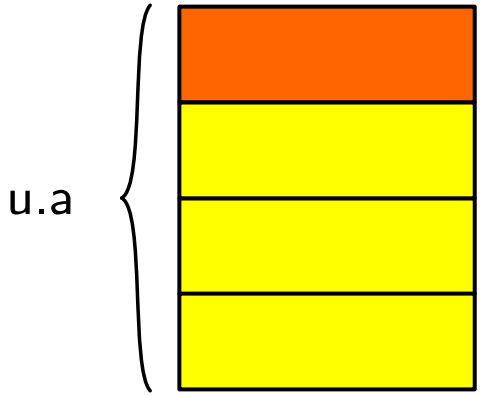


# Uniones
## Concepto

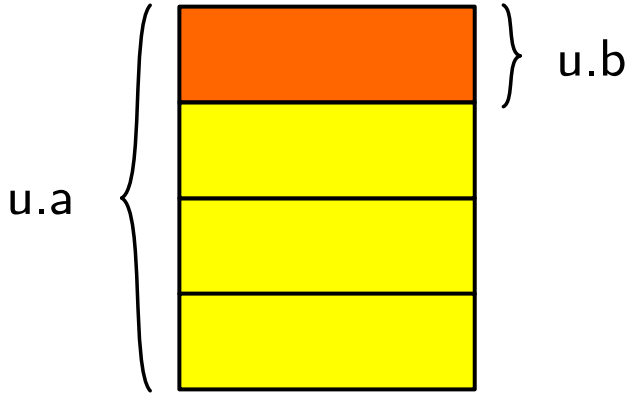
# Uniones
## Concepto



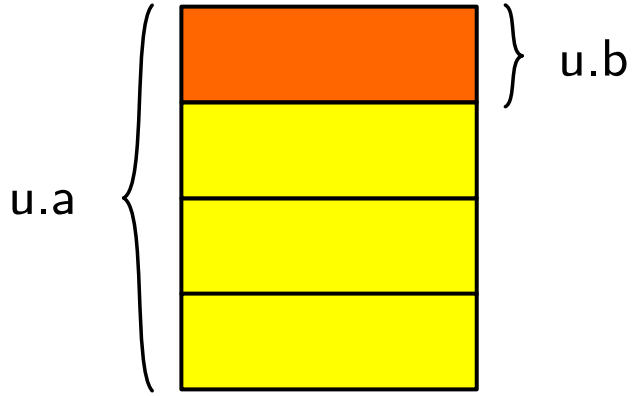
# Uniones
## Concepto



# Uniones
## Concepto

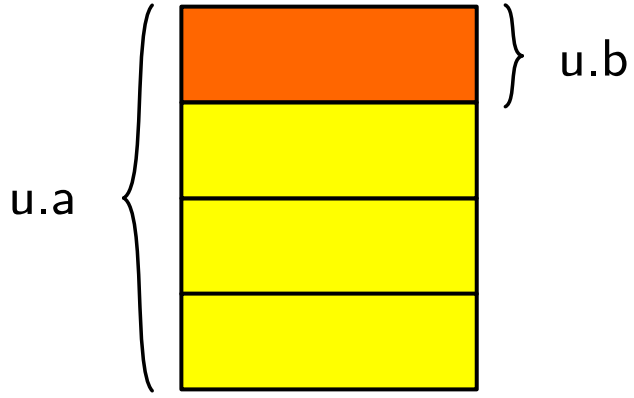


# Uniones
## Concepto




- El tamaño en memoria de la unión es la necesaria para almacenar el miembro que ocupe mayor espacio

# Uniones
## Concepto




- El tamaño en memoria de la unión es la necesaria para almacenar el miembro que ocupe mayor espacio
- Al compartir memoria, el acceso a los miembros podría tener resultados inesperados

# Uniones
## Definición

# Uniones
## Definición

```
union etiqueta {
   tipo1 nombre1;
   tipo2 nombre2;
         ⋮
   tipon nombren;
};
```

# Uniones
## Definición

```
union etiqueta {
   tipo1 nombre1;
   tipo2 nombre2;
         ⋮
   tipon nombren;
};
```

- se declara y define con la palabra clave `union`

# Uniones
## Definición

```
union etiqueta {
   tipo1 nombre1;
   tipo2 nombre2;
         ⋮
   tipon nombren;
};
```

- se declara y define con la palabra clave `union`
- *nombre*1, *nombre*2, ... *nombre*n son miembros de la unión

# Uniones
## Definición

```
union etiqueta {
   tipo1 nombre1;
   tipo2 nombre2;
         ⋮
   tipon nombren;
};
```

- se declara y define con la palabra clave `union`
- *nombre*1, *nombre*2, . . . *nombre*n son miembros de la unión
- *tipo*1, *tipo*2, . . . *tipo*n son los tipos de datos de los miembros de la unión, pueden ser diferentes

# Uniones
## Definición

```
union etiqueta {
   tipo1 nombre1;
   tipo2 nombre2;
         ⋮
   tipon nombren;
};
```

- se declara y define con la palabra clave `union`
- *nombre*1, *nombre*2, . . . *nombre*n son miembros de la unión
- *tipo*1, *tipo*2, . . . *tipo*n son los tipos de datos de los miembros de la unión, pueden ser diferentes
- la definición de la unión debe terminar con punto y coma

# Uniones
## Definición

```
union etiqueta {
   tipo1 nombre1;
   tipo2 nombre2;
         ⋮
   tipon nombren;
};
```

- se declara y define con la palabra clave `union`
- *nombre*1, *nombre*2, . . . *nombre*n son miembros de la unión
- *tipo*1, *tipo*2, . . . *tipo*n son los tipos de datos de los miembros de la unión, pueden ser diferentes
- la definición de la unión debe terminar con punto y coma
- al igual que en la estructura cada definición de los miembros de la unión debe terminar en punto y coma

# Uniones
## Acceso a miembros

# Uniones
## Acceso a miembros

```c
/* union-01.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-01.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-01.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-01.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-01.c
> ./a.out
* Asignando a int
  imprimiendo int => 42
```

# Uniones
## Acceso a miembros

# Uniones
## Acceso a miembros

```c
/* union-02.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;
  union int_float *pu = &u;

  printf("* Asignando a int\n");
  pu->entero = 42;
  printf("  imprimiendo int => %d\n", pu->entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-02.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;
  union int_float *pu = &u;

  printf("* Asignando a int\n");
  pu->entero = 42;
  printf("  imprimiendo int => %d\n", pu->entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-02.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;
  union int_float *pu = &u;

  printf("* Asignando a int\n");
  pu->entero = 42;
  printf("  imprimiendo int => %d\n", pu->entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-02.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;
  union int_float *pu = &u;

  printf("* Asignando a int\n");
  pu->entero = 42;
  printf("  imprimiendo int => %d\n", pu->entero);

  return 0;
}
```

# Uniones
## Acceso a miembros

```c
/* union-02.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;
  union int_float *pu = &u;

  printf("* Asignando a int\n");
  pu->entero = 42;
  printf("  imprimiendo int => %d\n", pu->entero);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-02.c
> ./a.out
* Asignando a int
  imprimiendo int => 42
```

# Uniones
## Uso

# Uniones
## Uso

```c
/* union-03.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Uso

```c
/* union-03.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Uso

```c
/* union-03.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a int\n");
  u.entero = 42;
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-03.c
> ./a.out
* Asignando a int
  imprimiendo int => 42
imprimiendo float => 0.00
```

# Uniones
## Uso

# Uniones
## Uso

```c
/* union-04.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a float\n");
  u.real = 42.0;
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Uso

```c
/* union-04.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a float\n");
  u.real = 42.0;
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Uso

```c
/* union-04.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u;

  printf("* Asignando a float\n");
  u.real = 42.0;
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-04.c
> ./a.out
* Asignando a float
  imprimiendo int => 1109917696
imprimiendo float => 42.00
```

# Uniones
## Inicialización

# Uniones
## Inicialización

```c
/* union-05.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {42};

  printf("* Inicializando con un int\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Inicialización

```c
/* union-05.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {42};

  printf("* Inicializando con un int\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Inicialización

```c
/* union-05.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {42};

  printf("* Inicializando con un int\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-05.c
> ./a.out
* Inicializando con un int
  imprimiendo int => 42
imprimiendo float => 0.00
```

# Uniones
## Inicialización

# Uniones
## Inicialización

```c
/* union-06.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Inicialización

```c
/* union-06.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Inicialización

```c
/* union-06.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-06.c
> ./a.out
* Inicializando con un float
  imprimiendo int => 42
imprimiendo float => 0.00
```

# Uniones
## Inicialización

# Uniones
## Inicialización

- En una declaración, una unión puede inicializarse con un valor del **mismo tipo** que el **primer miembro** del la unión

# Uniones
## Inicialización (Ojo solo c99 o >)

# Uniones
## Inicialización (Ojo solo c99 o >)

```c
/* union-07.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {.real=42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones

## Inicialización (Ojo solo c99 o >)

```c
/* union-07.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {.real=42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones
## Inicialización (Ojo solo c99 o >)

```c
/* union-07.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {.real=42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c99 --pedantic-errors union-07.c
> ./a.out
* Inicializando con un float
  imprimiendo int => 1109917696
imprimiendo float => 42.00
```

# Uniones

## Inicialización (Ojo solo c99 o >)

```c
/* union-07.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {.real=42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

# Uniones

## Inicialización (Ojo solo c99 o >)

```c
/* union-07.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u = {.real=42.0};

  printf("* Inicializando con un float\n");
  printf("  imprimiendo int => %d\n", u.entero);
  printf("imprimiendo float => %.2f\n", u.real);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-07.c
union-07.c: In function 'main':
union-07.c:11:24: error: ISO C90 forbids specifying
    subobject to initialize [-Wpedantic]
   union int_float u = {.real=42.0};
                       ^
```

# Uniones
## Operaciones permitidas

# Uniones
## Operaciones permitidas

### Asignación de una unión a otra del mismo tipo

```c
/* union-08.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42};
  union int_float u2;

  printf("* Asignando a unión\n");
  u2 = u1;
  printf("  imprimiendo int => %d\n", u2.entero);

  return 0;
}
```

# Uniones
## Operaciones permitidas

### Asignación de una unión a otra del mismo tipo

```c
/* union-08.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42};
  union int_float u2;

  printf("* Asignando a unión\n");
  u2 = u1;
  printf("  imprimiendo int => %d\n", u2.entero);

  return 0;
}
```

# Uniones
## Operaciones permitidas

# Uniones
## Operaciones permitidas

### Uso del operador &

```c
/* union-09.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42};
  union int_float *pu;

  printf("* Asignando a unión\n");
  pu = &u1;
  printf("  imprimiendo int => %d\n", (*pu).entero);

  return 0;
}
```

# Uniones
## Operaciones permitidas

### Uso del operador &

```c
/* union-09.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42};
  union int_float *pu;

  printf("* Asignando a unión\n");
  pu = &u1;
  printf("  imprimiendo int => %d\n", (*pu).entero);

  return 0;
}
```

# Uniones
## Operaciones no permitidas

# Uniones
## Operaciones no permitidas

### Operador == y !=

```c
/* union-10.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42}, u2 = {42};

  if ( u1 == u2 ) printf("Son iguales\n");

  return 0;
}
```

# Uniones
## Operaciones no permitidas

### Operador == y !=

```c
/* union-10.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42}, u2 = {42};

  if ( u1 == u2 ) printf("Son iguales\n");

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-10.c
union-10.c: In function 'main':
union-10.c:13:11: error: invalid operands to binary == (have
     'union int_float' and 'union int_float')
   if ( u1 == u2 ) printf("Son iguales\n");
           ^~
```

# Uniones
## Operaciones no permitidas

# Uniones

## Operaciones no permitidas

### Operador + y -

```c
/* suma.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42}, u2 = {99}, u3;

  u3 = u1 + u2;
  printf("  imprimiendo int => %d\n", u3.entero);

  return 0;
}
```

# Uniones

## Operaciones no permitidas

### Operador + y -

```c
/* suma.c */
#include <stdio.h>

union int_float {
  int entero;
  float real;
};

int main(void)
{
  union int_float u1 = {42}, u2 = {99}, u3;

  u3 = u1 + u2;
  printf("  imprimiendo int => %d\n", u3.entero);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors suma.c
suma.c: In function 'main':
suma.c:13:11: error: invalid operands to binary +
                (have 'union int_float' and 'union int_float')
   u3 = u1 + u2;
           ^
```

# Uniones
## Uso en general

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)

```
union int_4char {
  int entero;
  char c[4];
};
```

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones

```
union int_float {
  int entero;
  float real;
};

union int_float u[10];
```

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros

```
union numero {
  int entero;
  float real;
  struct {
    short int r;
    short int i;
  } complejo;
};
```

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros
- y las estructuras pueden tener uniones como miembros

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros
- y las estructuras pueden tener uniones como miembros

```
struct int_float {
  int es_entero;
  union {
    int entero;
    float real;
  } num;
};


if ( n.es_entero == 1 )
  printf("  imprimiendo int => %d\n", n.num.entero );
else
  printf("  imprimiendo float => %.2f\n", n.num.real );
```

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros
- y las estructuras pueden tener uniones como miembros

# Uniones
## Uso en general

- Las uniones pueden tener arreglos como miembros (los cuales son pasados por valor a las funciones)
- Se pueden hacer arreglos de uniones
- Las uniones pueden tener estructuras como miembros
- y las estructuras pueden tener uniones como miembros
- o cualquier combinación de lo anterior

# Uniones
## Ejemplos de uso

# Uniones
## Ejemplos de uso

```c
/* union-13.c */
#include <stdio.h>

struct int_float {
  char es_entero;
  union {
    int entero;
    float real;
  } num;
};

int main(void)
{
  struct int_float n = {1, {42}};

  n.es_entero = 0;
  n.num.real = 42.0;

  if ( n.es_entero )
    printf("  imprimiendo int => %d\n", n.num.entero );
  else
    printf("  imprimiendo float => %.2f\n", n.num.real );

  return 0;
}
```

# Uniones
## Ejemplos de uso

# Uniones

## Ejemplos de uso

```c
/* union-14.c */
#include <stdio.h>
#include <string.h>

union mensajes {
  char info[80];
  char aviso[80];
  char error[80];
};

int main(void)
{
  union mensajes msg;

  strcpy(msg.info, "Uso para ahorro de memoria");
  printf("%s\n", msg.info);
  printf("Uso de memoria: %lu bytes\n", sizeof msg);

  return 0;
}
```

# Uniones
## Ejemplos de uso

```c
/* union-14.c */
#include <stdio.h>
#include <string.h>

union mensajes {
  char info[80];
  char aviso[80];
  char error[80];
};

int main(void)
{
  union mensajes msg;

  strcpy(msg.info, "Uso para ahorro de memoria");
  printf("%s\n", msg.info);
  printf("Uso de memoria: %lu bytes\n", sizeof msg);

  return 0;
}
```

```
> gcc -Wall -ansi -std=c90 --pedantic-errors union-14.c
> ./a.out
Uso para ahorro de memoria
Uso de memoria: 80 bytes
```

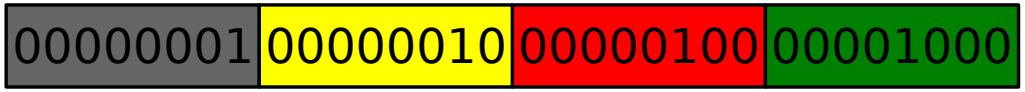
00000000 00000000 00000000 00000000

00000001 00000000 00000000 00000000

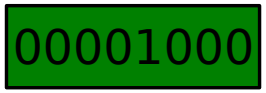
00000001 00000010 00000000 00000000

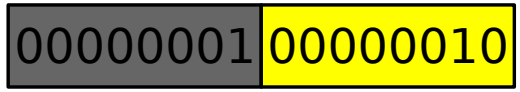
00000001 00000010 00000100 00000000

00000001 00000010 00000100 00001000

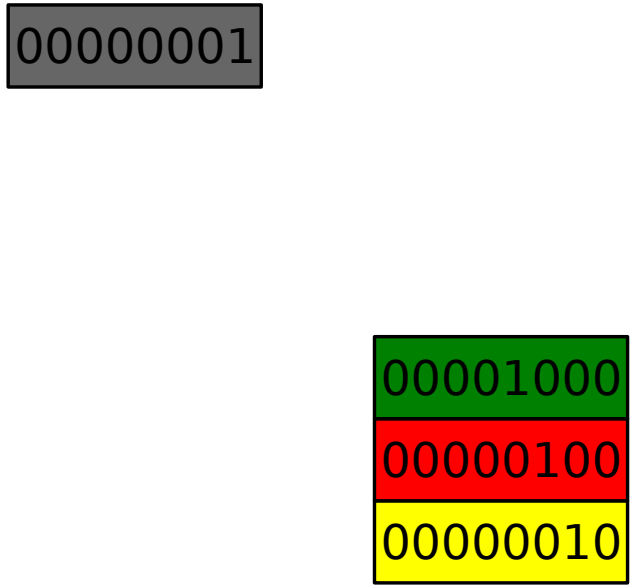

00000001
00000010
00000100
00001000

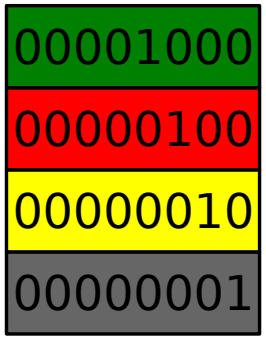

00000001
00000010
00000100


00001000

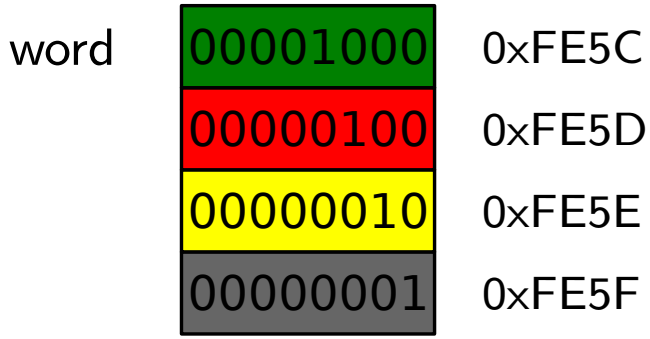

00000001
00000010


00001000
00000100

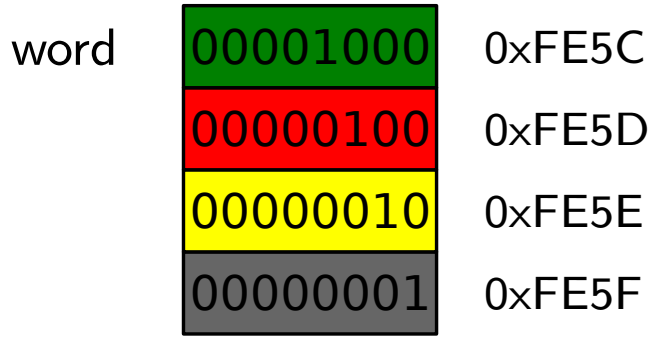

00000001
00001000
00000100
00000010


00001000
00000100
00000010
00000001


00001000
0xFE5C
00000100
0xFE5D
00000010
0xFE5E
00000001
0xFE5F




```
int word = 0x01020408;

/*b 00000001 00000010 00000100 00001000 */
/*d          1          2          4          8 */
/*x     0    1     0    2     0    4     0    8 */

printf("%d\n", word);
```

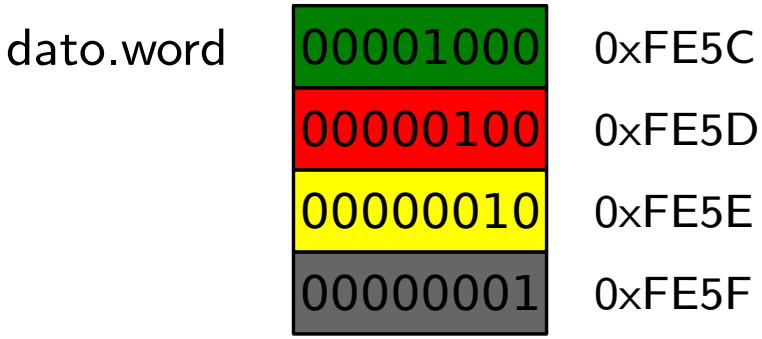



```c
int word = 0x01020408;

/*b 00000001 00000010 00000100 00001000 */
/*d          1          2          4          8 */
/*x     0    1     0    2     0    4     0    8 */

printf("%d\n", word);
```

```
16909320
```

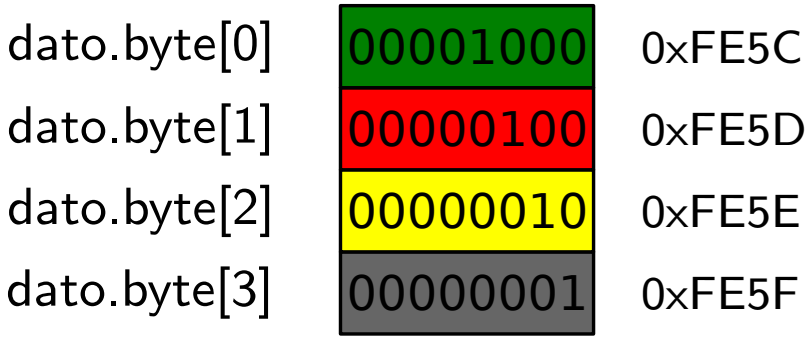

00001000
0xFE5C
00000100
0xFE5D
00000010
0xFE5E
00000001
0xFE5F


```
union int_4byte {
  unsigned int word;
  unsigned char byte[4];
};

union int_4byte dato;
```




```
union int_4byte {
  unsigned int word;
  unsigned char byte[4];
};

union int_4byte dato;
```


```
union int_4byte {
  unsigned int word;
  unsigned char byte[4];
};

union int_4byte dato;
```




```
union int_4byte {
  unsigned int word;
  unsigned char byte[4];
};

union int_4byte dato;
```

# Uniones
## Ejemplos de uso

# Uniones

## Ejemplos de uso

```c
/* union-15.c */
#include <stdio.h>

union int_4byte {
  unsigned int word;
  unsigned char byte[4];
};

int main(void)
{
  union int_4byte dato = {0x01020408};

  printf("%p %d\n", (void*) &dato.word, dato.word);
  printf("%p %d\n", (void*) &dato.byte[0], dato.byte[0]);
  printf("%p %d\n", (void*) &dato.byte[1], dato.byte[1]);
  printf("%p %d\n", (void*) &dato.byte[2], dato.byte[2]);
  printf("%p %d\n", (void*) &dato.byte[3], dato.byte[3]);

  return 0;
}
```

# Uniones

## Ejemplos de uso

```c
/* union-15.c */
#include <stdio.h>

union int_4byte {
  unsigned int word;
  unsigned char byte[4];
};

int main(void)
{
  union int_4byte dato = {0x01020408};

  printf("%p %d\n", (void*) &dato.word, dato.word);
  printf("%p %d\n", (void*) &dato.byte[0], dato.byte[0]);
  printf("%p %d\n", (void*) &dato.byte[1], dato.byte[1]);
  printf("%p %d\n", (void*) &dato.byte[2], dato.byte[2]);
  printf("%p %d\n", (void*) &dato.byte[3], dato.byte[3]);

  return 0;
}
```

```
0x7ffc11b7738c 16909320
0x7ffc11b7738c 8
0x7ffc11b7738d 4
0x7ffc11b7738e 2
0x7ffc11b7738f 1
```

# Uniones
## Ejemplos de uso

# Uniones
## Ejemplos de uso

# Uniones
## Ejemplos de uso

fragmento de `main.h` en el firmware del RoMAA-II

```
/**************************************
Uniones y Variables para Recepcion y Envio */
typedef union {
  unsigned char uc_data[4];
  float f_data;
} uchar_to_float_t;

typedef union {
  unsigned char uc_data[4];
  int i_data;
} uchar_to_int_t;
```

# Uniones
## Ejemplos de uso

# Uniones
## Ejemplos de uso

fragmento de `main.c` en el firmware del RoMAA-II

```
uchar_to_float_t uchar_to_float;
uchar_to_int_t uchar_to_int;
```

# Uniones
## Ejemplos de uso

# Uniones
## Ejemplos de uso

fragmento de `main.c` en el firmware del RoMAA-II

```
uchar_to_float.uc_data[0] = comando[1];
uchar_to_float.uc_data[1] = comando[2];
uchar_to_float.uc_data[2] = comando[3];
uchar_to_float.uc_data[3] = comando[4];
x = uchar_to_float.f_data;
```

# Uniones
## Ejemplos de uso

# Uniones
## Ejemplos de uso

```c
/* union-16.c */
#include <stdio.h>

union float_4byte {
  float real;
  unsigned char byte[4];
};

typedef unsigned char uchar;

float bytes2float (uchar c0, uchar c1, uchar c2, uchar c3)
{
  union float_4byte dato;

  dato.byte[0] = c0;
  dato.byte[1] = c1;
  dato.byte[2] = c2;
  dato.byte[3] = c3;

  return dato.real;
}
```

# Uniones
## Ejemplos de uso

```
/* continúa union-16.c */

int main(void)
{
  union float_4byte cmd = {23.5};
  float recibido;

  recibido = bytes2float(cmd.byte[0], cmd.byte[1],
                         cmd.byte[2], cmd.byte[3]);

  printf("%.2f\n", recibido);

  return 0;
}
```

# Uniones
## Ejemplos de uso

```
/* continúa union-16.c */

int main(void)
{
  union float_4byte cmd = {23.5};
  float recibido;

  recibido = bytes2float(cmd.byte[0], cmd.byte[1],
                         cmd.byte[2], cmd.byte[3]);

  printf("%.2f\n", recibido);

  return 0;
}
```

```
23.50
```

# Uniones
## Uso en funciones

# Uniones
## Uso en funciones

### Paso a función por valor

```
void fusaunion(union int_float u)
{
  u.entero = 42;
  printf("  imprimiendo en función => %d\n", u.entero);
}
```

### Retorno desde función

```
union int_float fdevunion(void)
{
  union int_float u = {99};
  return u;
}
```

# Uniones
## Uso en funciones

# Uniones
## Uso en funciones

### Cuidado con las uniones con arreglos como miembros

```c
/* union-12.c */
#include <stdio.h>
#include <string.h>

union int_4char {
  int entero;
  char c[4];
};

void fusaunion(union int_4char u)
{
  strcpy(u.c, "HAL");
  printf("  imprimiendo char => %c\n", u.c[0]);
}
```

# Uniones
## Uso en funciones

### Cuidado con las uniones con arreglos como miembros

```c
/* Continúa union-12.c */

int main(void)
{
  union int_4char u;

  printf("* Asignando a int\n");
  u.entero = 42;
  fusaunion(u);
  printf("  imprimiendo int => %d\n", u.entero);

  return 0;
}
```

# Uniones
## Uso en funciones

### Cuidado con las uniones con arreglos como miembros

```c
/* Continúa union-12.c */

int main(void)
{
  union int_4char u;

  printf("* Asignando a int\n");
  u.entero = 42;
  fusaunion(u);
  printf("  imprimiendo int => %d\n", u.entero);

  return 0;
}
```

```
    * Asignando a int
  imprimiendo char => H
  imprimiendo int => 42
```

# Consultas

claudiojpaz@gmail.com

Horario de Consulta: Lunes 17:00-19:00hs
Of.5 Ed.Salcedo