



LABORATORIS

PROGRAMACIÓ CIENTÍFICA

Sessió 9: Registres
i tipus d'usuari


REGISTRES

I TIPUS D'USUARI

Structs

- Una estructura és una col·lecció d'una o més variables, de tipus possiblement diferents, agrupades sota un mateix nom perquè sigui més convenient de tractar. [En altres llenguatges = “records”]
- Ajuden a organitzar dades complicades perquè és millor tenir-ho agrupat que per separat

Declaració d'estructures en C:



```
struct {  
    int x;  
    int y;  
} punt;
```

Structs

- Una estructura és una col·lecció d'una o més variables, de tipus possiblement diferents, agrupades sota un mateix nom perquè sigui més convenient de tractar. [En altres llenguatges = "records"]
- Ajuden a organitzar dades complicades perquè és millor tenir-ho agrupat que per separat

Declaració d'estructures en C:

Paraula reservada:
struct

```
struct {  
    int x;  
    int y;  
} punt;
```

Camps de la meva
estructura/registre

Nom de la meva
variable

Structs

- Una estructura és una col·lecció d'una o més variables, de tipus possiblement diferents, agrupades sota un mateix nom perquè sigui més convenient de tractar. [En altres llenguatges = "records"]
- Ajuden a organitzar dades complicades perquè és millor tenir-ho agrupat que per separat

Declaració d'estructures en C:

- Aquesta sentència defineix una única variable, anomenada "punt", que té dos camps: x i y.

Paraula reservada:
struct

```
struct {  
    int x;  
    int y;  
} punt;
```

Camps de la meva
estructura/registre

Nom de la meva
variable

Structs

- Una estructura és una col·lecció d'una o més variables, de tipus possiblement diferents, agrupades sota un mateix nom perquè sigui més convenient de tractar. [En altres llenguatges = "records"]
- Ajuden a organitzar dades complicades perquè és millor tenir-ho agrupat que per separat

Declaració d'estructures en C:

Paraula reservada:
struct

```
struct {  
    int x;  
    int y;  
} punt;
```

Camps de la meua
estructura/registre

Nom de la meua
variable

- Aquesta sentència defineix una única variable, anomenada "punt", que té dos camps: x i y.
- És equivalent a declarar qualsevol altra variable

Per exemple:

int a;

float b;

struct { int x; int y; } punt;

Structs

- Una estructura és una col·lecció d'una o més variables, de tipus possiblement diferents, agrupades sota un mateix nom perquè sigui més convenient de tractar. [En altres llenguatges = "records"]
- Ajuden a organitzar dades complicades perquè és millor tenir-ho agrupat que per separat

Declaració d'estructures en C:

Paraula reservada:
struct

```
struct {  
    int x;  
    int y;  
} punt;
```

Camps de la meua
estructura/registre

Nom de la meua
variable

- Aquesta sentència defineix una única variable, anomenada "punt", que té dos camps: x i y.
- És equivalent a declarar qualsevol altra variable

Per exemple:

int a;

float b;

struct { int x; int y; } punt;

- Si necessito una altra variable del mateix tipus, he de tornar a escriure l'estructura (no repetir codi!):

struct { int x; int y; } punt1;


struct { int x; int y; } punt2;



Structs

Declaració com a tipus d'usuari

- Per reaprofitar una estructura, puc fer-ho afegint-hi un nom, anomenat “**structure tag**” (rótulo de estructura).
- Si declaro un “structure tag”, podré fer servir aquest nom per referir-me a l'estructura, i no caldrà tornar a escriure l'estructura (i repetir codi, que no s'ha de fer)



```
struct Punt{  
    int x;  
    int y;  
};  
  
struct Punt p1 = {200,300};
```

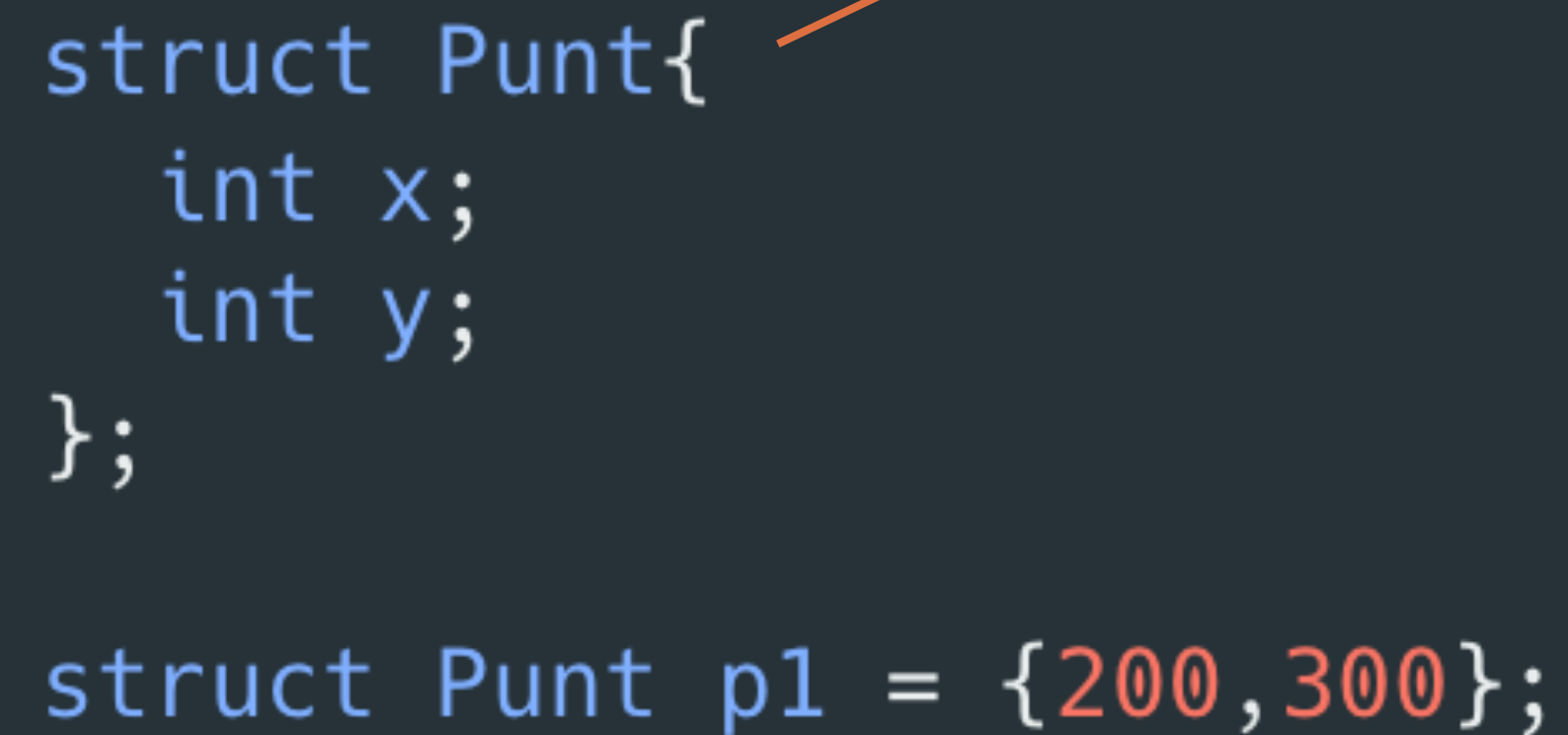

Structs

Declaració com a tipus d'usuari

- Per reaprofitar una estructura, puc fer-ho afegint-hi un nom, anomenat “**structure tag**” (rótulo de estructura).
- Si declaro un “structure tag”, podré fer servir aquest nom per referir-me a l'estructura, i no caldrà tornar a escriure l'estructura (i repetir codi, que no s'ha de fer)

Structure Tag:

Escriu “Punt” començant en majúscula per destacar que és un tipus.



```
struct Punt{  
    int x;  
    int y;  
};  
  
struct Punt p1 = {200,300};
```

Structs

Declaració com a tipus d'usuari

- Per reaprofitar una estructura, puc fer-ho afegint-hi un nom, anomenat “**structure tag**” (rótulo de estructura).
- Si declaro un “structure tag”, podré fer servir aquest nom per referir-me a l'estructura, i no caldrà tornar a escriure l'estructura (i repetir codi, que no s'ha de fer)

Structure Tag:

Escriu “Punt” començant en majúscula per destacar que és un tipus.

```
struct Punt{  
    int x;  
    int y;  
};
```

De moment no he declarat cap variable de tipus struct Punt

```
struct Punt p1 = {200,300};
```

Structs

Declaració com a tipus d'usuari

- Per reaprofitar una estructura, puc fer-ho afegint-hi un nom, anomenat “**structure tag**” (rótulo de estructura).
- Si declaro un “structure tag”, podré fer servir aquest nom per referir-me a l'estructura, i no caldrà tornar a escriure l'estructura (i repetir codi, que no s'ha de fer)

Structure Tag:

Escric “Punt” començant en majúscula per destacar que és un tipus.

```
struct Punt{  
    int x;  
    int y;  
};
```

De moment no he declarat cap variable de tipus struct Punt

```
struct Punt p1 = {200, 300};
```

Aquí declaro una variable p1 de tipus struct Punt i l'inicialitzo

Structs

Declaració amb typedef

- Per evitar haver d'escriure "struct Punt" cada cop, puc crear un alias amb **typedef**.

```
typedef existing_type new_type;
```

Structs

Declaració amb typedef

- Per evitar haver d'escriure "struct Punt" cada cop, puc crear un alias amb **typedef**.

```
typedef existing_type new_type;
```

Per exemple:

```
typedef int Enter;
```

Structs

Declaració amb typedef


- Per evitar haver d'escriure "struct Punt" cada cop, puc crear un alias amb **typedef**.

```
typedef existing_type new_type;
```

Per exemple:

```
typedef int Enter;
```

```
typedef struct Punt{ int x; int y;} Punt;
```



```
typedef struct Punt{  
    int x;  
    int y;  
} Punt;
```

```
Punt p1 = {200, 300};
```


Structs

Accés als camps/membres

- Per accedir a cadascun dels camps: farem servir el punt (.)
- Escriviu aquest codi i comproveu que funciona.



```
#include <stdio.h>

typedef struct Punt{
    int x;
    int y;
} Punt;

int main(){

    Punt p1 = {200,300};

    printf("Les coordenades del punt son: (%d, %d)\n", p1.x, p1.y);

    return 0;
}
```

Les coordenades del punt son: (200, 300)

Structs

Accés als camps/membres

- Per accedir a cadascun dels camps: farem servir el punt (.)
- Escriviu aquest codi i comproveu que funciona.



```
#include <stdio.h>

typedef struct Punt{
    int x;
    int y;
} Punt;

int main(){

    Punt p1 = {200,300};

    printf("Les coordenades del punt son: (%d, %d)\n", p1.x, p1.y);

    p1.x = 111;
    p1.y = 222;

    printf("Les coordenades del punt son: (%d, %d)\n", p1.x, p1.y);

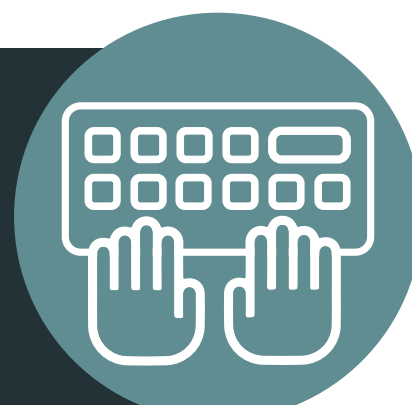
    return 0;
}
```

Les coordenades del punt son: (200, 300)
Les coordenades del punt son: (111, 222)

Structs

Com a paràmetres de funcions

- Com que hem definit un nou tipus, especifiquem que el paràmetre serà d'aquell tipus i ja està.
- Escriviu la funció **imprimir_punt**



```
#include <stdio.h>

typedef struct Punt{
    int x;
    int y;
} Punt;

void imprimir_punt(Punt p){
    printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
}

int main(){

    Punt p1 = {200,300};

    imprimir_punt(p1);
    p1.x = 111;
    p1.y = 222;
    imprimir_punt(p1);

    return 0;
}
```

```
Les coordenades del punt son: (200, 300)
Les coordenades del punt son: (111, 222)
```


Structs

Com a paràmetres de funcions

- Com que hem definit un nou tipus, especifiquem que el paràmetre serà d'aquell tipus i ja està.

El pas de paràmetres es fa per valor

- Encara que siguin estructures, el pas de paràmetres és per valor igualment.

```
#include <stdio.h>

typedef struct Punt{
    int x;
    int y;
} Punt;

void intent_de_modificar_punt(Punt p){
    p.x = 666;
    p.y = 666;
}

void imprimir_punt(Punt p){
    printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
}

int main(){

    Punt p1 = {200,300};
    intent_de_modificar_punt(p1);
    imprimir_punt(p1); // No s'ha modificat perquè pas per valor

    return 0;
}
```

Les coordenades del punt son: (200, 300)

Structs

Com a paràmetres de funcions

- Com que hem definit un nou tipus, especifiquem que el paràmetre serà d'aquell tipus i ja està.

El pas de paràmetres es fa per valor

- Encara que siguin estructures, el pas de paràmetres és per valor igualment.

Retornar structs

- Podem definir una funció que retorna un struct

```
#include <stdio.h>

typedef struct Punt{
    int x;
    int y;
} Punt;

Punt suma_punts(Punt p1, Punt p2){
    Punt aux;
    aux.x = p1.x + p2.x;
    aux.y = p1.y + p2.y;
    return aux;
}

void imprimir_punt(Punt p){
    printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
}

int main(){

    Punt p1 = {200,300};
    Punt p2 = {10, 20};
    Punt p3;
    p3 = suma_punts(p1,p2);
    imprimir_punt(p3);

    return 0;
}
```

Les coordenades del punt son: (210, 320)

Structs

Util: funció "constructor"

- Podem crear una funció que ens "ompli" els camps d'un struct, per encapsular un procediment que farem moltes vegades

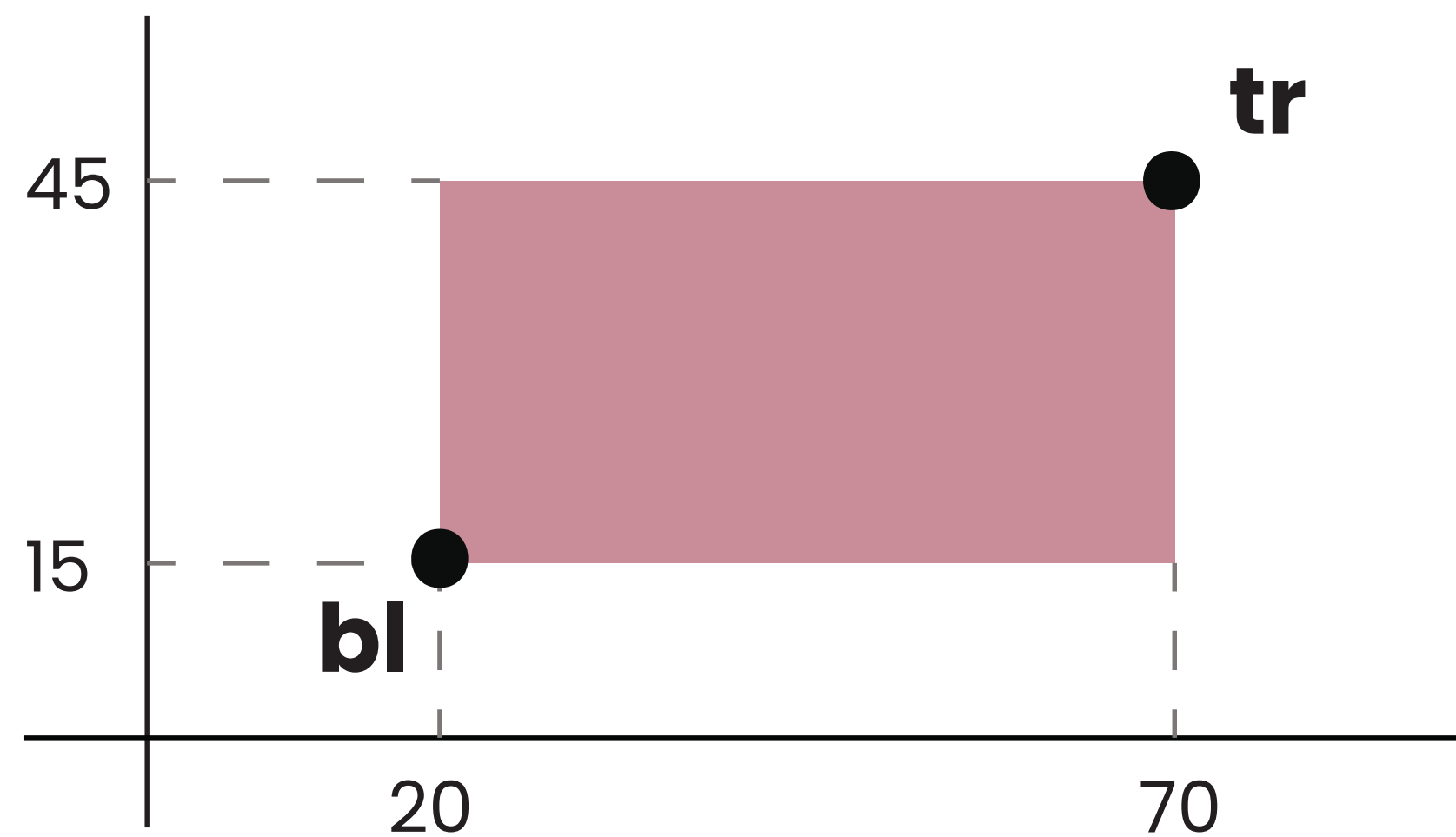
```
1 #include <stdio.h>
2
3 typedef struct Punt{
4     int x;
5     int y;
6 } Punt;
7
8 Punt constructor_punt(int x, int y){
9     Punt p;
10    p.x = x;
11    p.y = y;
12    return p;
13 }
14
15 void imprimir_punt(Punt p){
16    printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
17 }
18
19 int main(){
20
21    Punt p1;
22    p1 = constructor_punt(3,4);
23    imprimir_punt(p1);
24
25    return 0;
26 }
```

Les coordenades del punt son: (3, 4)

Structs

Structs anidats

- Podem tenir structs dins de structs



```
1 #include <stdio.h>
2
3 typedef struct Punt{
4     int x;
5     int y;
6 } Punt;
7
8 typedef struct Rectangle{
9     Punt bl; // Bottom-left corner
10    Punt tr; // Top-right corner
11 } Rectangle;
12
13 void print_rectangle(Rectangle r){
14     printf("Bottom left corner: (%d,%d)\n", r.bl.x, r.bl.y);
15     printf("Top right corner:   (%d,%d)\n", r.tr.x, r.tr.y);
16
17 }
18 int main(){
19
20     Rectangle r;
21     r.bl.x = 20;
22     r.bl.y = 15;
23     r.tr.x = 70;
24     r.tr.y = 45;
25
26     print_rectangle(r);
27
28     return 0;
29 }
```

Structs

Pas per referència

- Si necessitem modificar un punt dins una funció, necessitem passar-ho per referència.

```
1 #include <stdio.h>
2
3 typedef struct Punt{
4     int x;
5     int y;
6 } Punt;
7
8 void imprimir_punt(Punt p){
9     printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
10 }
11
12 // No funciona perquè és pas per valor
13 void incrementa_punt(Punt p){
14     p.x++;
15     p.y++;
16 }
17
18 int main(){
19
20     Punt p1 = {100, 200};
21     incrementa_punt(p1);
22     imprimir_punt(p1);
23
24     return 0;
25 }
```

Les coordenades del punt son: (100, 200)

Structs

Pas per referència

- Si necessitem modificar un punt dins una funció, necessitem passar-ho per referència.
- Per accedir als camps d'un struct si el passem per punter necessitem fer servir el parèntesi:

(*p).camp

```
1 #include <stdio.h>
2
3 typedef struct Punt{
4     int x;
5     int y;
6 } Punt;
7
8 void imprimir_punt(Punt p){
9     printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
10 }
11
12 // Pas per referència
13 void incrementa_punt(Punt *p){
14     (*p).x++;
15     (*p).y++;
16 }
17
18 int main(){
19
20     Punt p1 = {100, 200};
21     incrementa_punt(&p1);
22     imprimir_punt(p1);
23
24     return 0;
25 }
```

Les coordenades del punt son: (101, 201)

Structs

Pas per referència

- Si necessitem modificar un punt dins una funció, necessitem passar-ho per referència.
- Per accedir als camps d'un struct si el passem per punter necessitem fer servir el parèntesi:

(*p).camp

```
1 #include <stdio.h>
2
3 typedef struct Punt{
4     int x;
5     int y;
6 } Punt;
7
8 void imprimir_punt(Punt p){
9     printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
10 }
11
12 // Pas per referència
13 void incrementa_punt(Punt *p){
14     (*p).x++;
15     (*p).y++;
16 }
17
18 int main(){
19
20     Punt p1 = {100, 200};
21     incrementa_punt(&p1);
22     imprimir_punt(p1);
23
24     return 0;
25 }
```

Les coordenades del punt son: (101, 201)

Structs

Pas per referència

- Si necessitem modificar un punt dins una funció, necessitem passar-ho per referència.
- Per accedir als camps d'un struct si el passem per punter necessitem fer servir el parèntesi:

(*p).camp

- Manera alternativa:

p -> camp



```
1 #include <stdio.h>
2
3 typedef struct Punt{
4     int x;
5     int y;
6 } Punt;
7
8 void imprimir_punt(Punt p){
9     printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
10 }
11
12 // Accés alternatiu als camps
13 void incrementa_punt(Punt *p){
14     p->x++;
15     p->y++;
16 }
17
18 int main(){
19
20     Punt p1 = {100, 200};
21     incrementa_punt(&p1);
22     imprimir_punt(p1);
23
24     return 0;
25 }
```

Les coordenades del punt son: (101, 201)

Taules de structs

```

#include <stdio.h>
#define NUM_PUNTS 10

typedef struct Punt{
    int x;
    int y;
} Punt;

void imprimir_punt(Punt p){
    printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
}

int main(){

    Punt taula_punts[NUM_PUNTS];

    for (int i=0; i<NUM_PUNTS; i++){
        taula_punts[i].x = (i+1);
        taula_punts[i].y = (i+1)*10;
        imprimir_punt(taula_punts[i]);
    }
    return 0;
}
```

Taules de structs

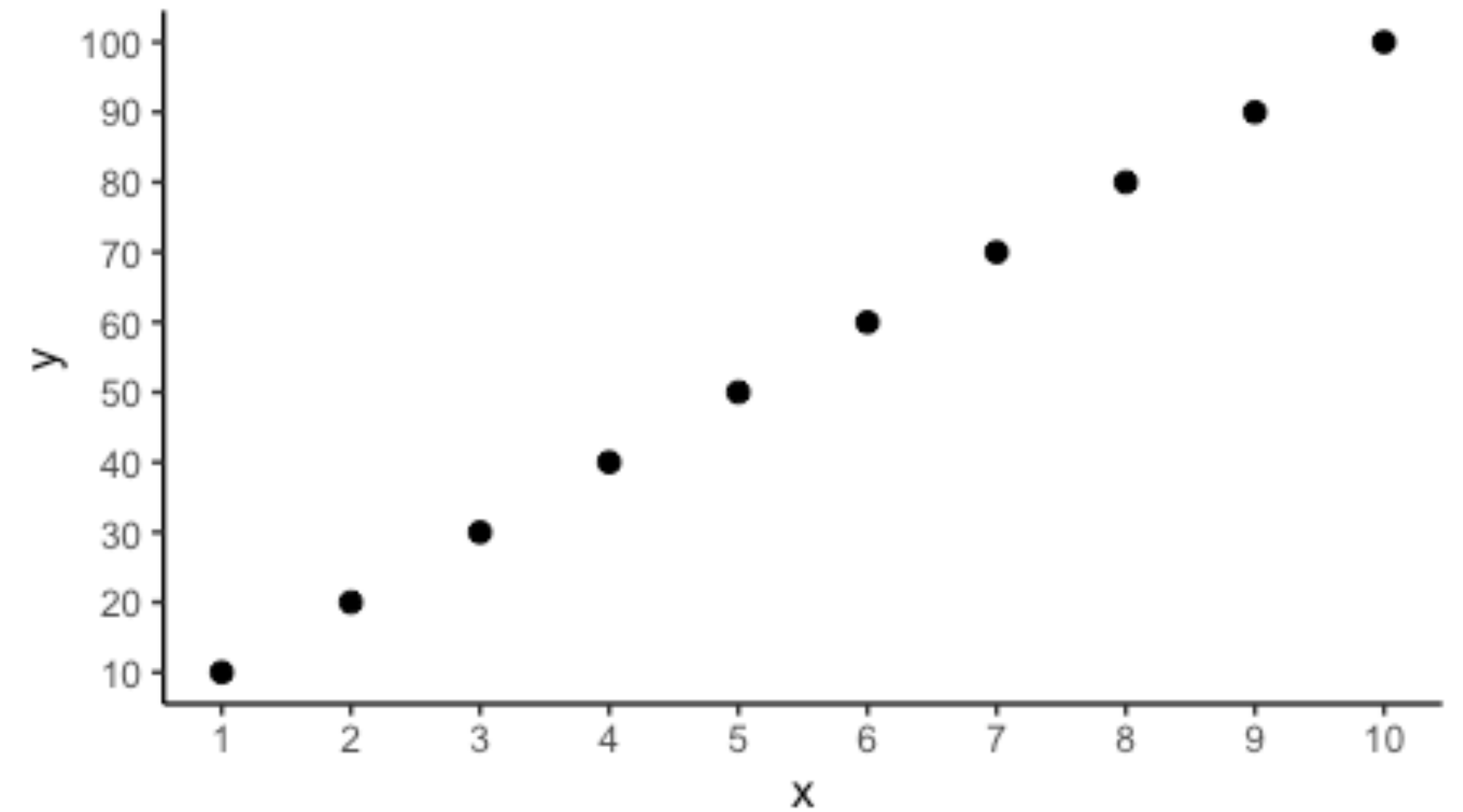
```
#include <stdio.h>
#define NUM_PUNTS 10

typedef struct Punt{
    int x;
    int y;
} Punt;

void imprimir_punt(Punt p){
    printf("Les coordenades del punt son: (%d, %d)\n", p.x, p.y);
}

int main(){
    Punt taula_punts[NUM_PUNTS];

    for (int i=0; i<NUM_PUNTS; i++){
        taula_punts[i].x = (i+1);
        taula_punts[i].y = (i+1)*10;
        imprimir_punt(taula_punts[i]);
    }
    return 0;
}
```



Les coordenades del punt son: (1, 10)
Les coordenades del punt son: (2, 20)
Les coordenades del punt son: (3, 30)
Les coordenades del punt son: (4, 40)
Les coordenades del punt son: (5, 50)
Les coordenades del punt son: (6, 60)
Les coordenades del punt son: (7, 70)
Les coordenades del punt son: (8, 80)
Les coordenades del punt son: (9, 90)
Les coordenades del punt son: (10, 100)

A CASA...

EXERCICIS L9 del Moodle

EL PROPER DIA...

UTILITATS AVANÇADES (I)