Introduktion till C-programmering

Erik Sintorn
Original slides by
Pedro Trancoso and Ulf Assarsson

Vad är erat favoritspråk?

Mentometerdags!

Den tidiga utvecklingen av programspråk

1949: Short Code, 1:st high level language
1957: Fortran, IBM
1958: Lisp
1960: Cobol 60 (Common Business-oriented language)
1964: BASIC
1960: ALGOL 60 (ALGOrithmic Language 1960)
1960: Simula (Introduced classes)
1972: Prolog
1973: C
1975: Ada
1975: Pascal
1978: ML

1978,K&R C	(Kernighan & Ritchie)
1989,C89/C90	(ANSI-C)
1999,C99	(Rev. ANSI-standard)
2008, Embedded C	(fixed-point arithmetics, basic I/O operations)
2011,C11	(Rev. ANSI-standard)

Varför C, och inte ett modernt språk?

- Moderna Programspråk
  - C++, Java, javascript, C#, Objective-C, Rust, ...
- Maskinorienterad Programmering
  - Behöver kunna skriva till absoluta minnesadresser
  - Måste vara så snabbt som möjligt
  - Måste ofta vara extremt stabilt och säkert
- C är det absolut mest använda språket för MOP
  - Första språket på de flesta nya maskiner
  - Drivrutiner och annan maskinnära kod
  - och när ni kan C är ni mycket närmre att kunna C++

Varför C, och inte assembler?

- Portability
  - Vill inte skriva om koden för varje ny tamagoochi
- Enkelhet
- Performance (!)
  - Moderna kompilatorer är duktiga
- Kan alltid göra inline assembler

```
C:
a = b - c;

Assembler:
LDR   r3, =b
LDR   r2, [r3]
LDR   r3, =c
LDR   r3, [r3]
SUBS  r2, r2, r3
LDR    r3, =a
STR    r2, [r3]
```

C vs Java

- C har inga klasser, polymorphism, eller overloading
- C gör inga hemliga säkerhetskontroller åt dig:

```
int a[10];
a[100] = 123; // Java: "Katastrof!", C: "Okay.."
```

- Ingen exceptionhandling (try/catch)
- Implicita typkonverteringar
- Inga booleans
- Och mycket mera...
- C skall kompileras!

Hello World!

Eriks fusklapp:

- CodeLite (visa resurshemsidan)
- Starta CodeLite och bygg programmet
- Visa vart filerna hamnar
- Dubbelklicka (funkar inte)
- Kör från command line
- Kör från CodeLite
- Gå igenom programmet rad för rad.

Standardtyper och printf

```c
#include <stdio.h>

int main()
{
    short int si = 123;                 // Minimum 2 bytes (-32768 to 32767)
    unsigned short int usi = 123;       // Minimum 2 bytes (0 to 65535)
    int i = 123;                        // Minimum 4 bytes (-2,147,483,648 to 2,147,483,647)
    unsigned int ui = 123;              // Minimum 4 bytes (0 to 4,294,967,295)
    long long int lli = 123;            // Minimum 8 bytes (-(2^63) to (2^63)-1)
    float f = 123.123;                  // Minimum 4 bytes
    double d = 1234.1234;               // Minimum 8 bytes
    char c = 'a';                       // Minimum 1 bytes (-128 to 127)

    printf("i = %i, f = %f, c = %c\n", i, f, c);

    return 0;
}
```

Output:
i = 123, f = 123.123001, c = a

Deklarationer och Assignments

```c
#include <stdio.h>

int main()
{
    int a;                                  // Declaration of a
    int b;                                  // Declaration of b
    b = 123;                                // Assignement of b
    printf("a = %i, b = %i\n", a, b);
    return 0;
}
```

Output: ??

Typkonvertering

```c
#include <stdio.h>

int main()
{
    int a = 1;
    int b = 2;
    float c = a / b;                    // Implicit type conversion
    float d = ((float) a) / b;          // Explicit type conversion
    printf("c = %f, d = %f\n", c, d);
    return 0;
}
```

Output: c = 0, d = 0.5

Arrays och Vektorer

```c
#include <stdio.h>

int main()
{
    // Declaration and assignment of array
    int a[] = {1, 2, 3, 4, 5, 6};
    // Declaration of b (uninitialized)
    int b[10];
    // Assignement of element 3 in a
    a[3] = 3;

    // sizeof(...) returns the size in bytes
    int size_of_a = sizeof(a) / sizeof(int);

    // What is the output?
    for(int i=0; i<size_of_a; i++) {
        printf("%i", a[i]);
    }
    printf("\n");
    return 0;
}
```

Nope

Det finns inga dynamiska vektorer i språket eller standardbiblioteken.

Ni kan förstås implementera egna, eller använda 3rd party libs, men då måste man kunna dynamisk minnesallockering som inte är en del av språket.

Mer om sådant i en senare lektion.

Strängar

```c
#include <stdio.h>

int main()
{
    // Declaration and assignment of array
    char animal1[] = "Monkey";
    char animal2[] = {'M', 'o', 'n', 'k', 'e', 'y', 0 };
    char animal3[] = { 77, 111, 110, 107, 101, 121, 0 };
    printf("%s, %s, %s\n", animal1, animal2, animal3);
}
```

Output: Monkey, Monkey, Monkey

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

Funktioner

```c
#include <stdio.h>

int MakeUpNumber()
{
    return 5;
}
void PrintNumber(int number)
{
    printf("%d\n", number);
}
void ChangeNumber(int number)
{
    number = 4;                    // Changes only the local copy of number
}

int main()
{
    int a = MakeUpNumber();
    ChangeNumber(a);
    PrintNumber(a);                // Prints 5
}
```

Funktioner

```c
#include <stdio.h>

int MakeUpNumber()
{
    return 5;
}
void PrintNumber(int number)
{
    printf("%d\n", number);
}
void ChangeNumber(int number)
{
    number = 4;                    // Cha
}

int main()
{
    int a = MakeUpNumber();
    ChangeNumber(a);
    PrintNumber(a);                // Prints 5
}
```

```c
void Print(int number)
{
    printf("%i\n", number);
}
void Print(float number)
{
    printf("%f\n", number);
}
void Print(int number1, int number2)
{
    printf("%i, %i\n", number1, number2);
}
```

ERROR

C-funktioner

NOT AN ERROR!

```c
#include <stdio.h>

int MakeUpNumber()
{
    return 5;
}
void PrintNumber(int nu
{
    printf("%d\n", numb

void ChangeNumber(int n
{
    number = 4;
}

int main()
{
    int a = MakeUpNumbe
    ChangeNumber(a);
    PrintNumber(a);
}
```

```c
#include <stdio.h>

int CurrentHealth(int extra_health)
{
    int base_health = 5;
    if(extra_health > 0)
    {
        return base_health + extra_health;
    }
}

int main()
{
    printf("CurrentHealth: %i\n", CurrentHealth(5));
    printf("CurrentHealth: %i\n", CurrentHealth(0));
}
```

```c
void Print(int number)
{
    printf("%i\n", number);
}
```

Visibility

```c
#include <stdio.h>

int x;
void foo()
{
    // x is visible
    // y is not visibile
}

int y;
```

Visibility

```c
#include <stdio.h>

int x;
void foo()
{
    // x is visi
    // y is not
}

int y;
```

```c
#include <stdio.h>

char x;

void foo(float x)
{
    // argument x (float) is visible but NOT the (char) x
    if(x > 0.0) {
        int x = 5;
        // int x is visible, but NOT the function argument
        printf("%i\n", x);
    }
}
```

Visibility

```c
#include <stdio.h>

int x;
void foo()
{
    // x is visi
    // y is not
}

int y;
```

```c
#include <stdio.h>

char x;

void foo(float x)
{
    // argument x (float) is
    if(x > 0.0) {
        int x = 5;
        // int x is visible,
        printf("%i\n", x);
    }
}
```

```c
#include <stdio.h>

int x;

int foo(int x) {
    if( x == 0 ){
        int x = 4;
    }
    return x;
}

int main() {
    x = 1;
    x = foo(0);
    printf("x is %d\n", x);
    return 0;
}
```

Programstruktur

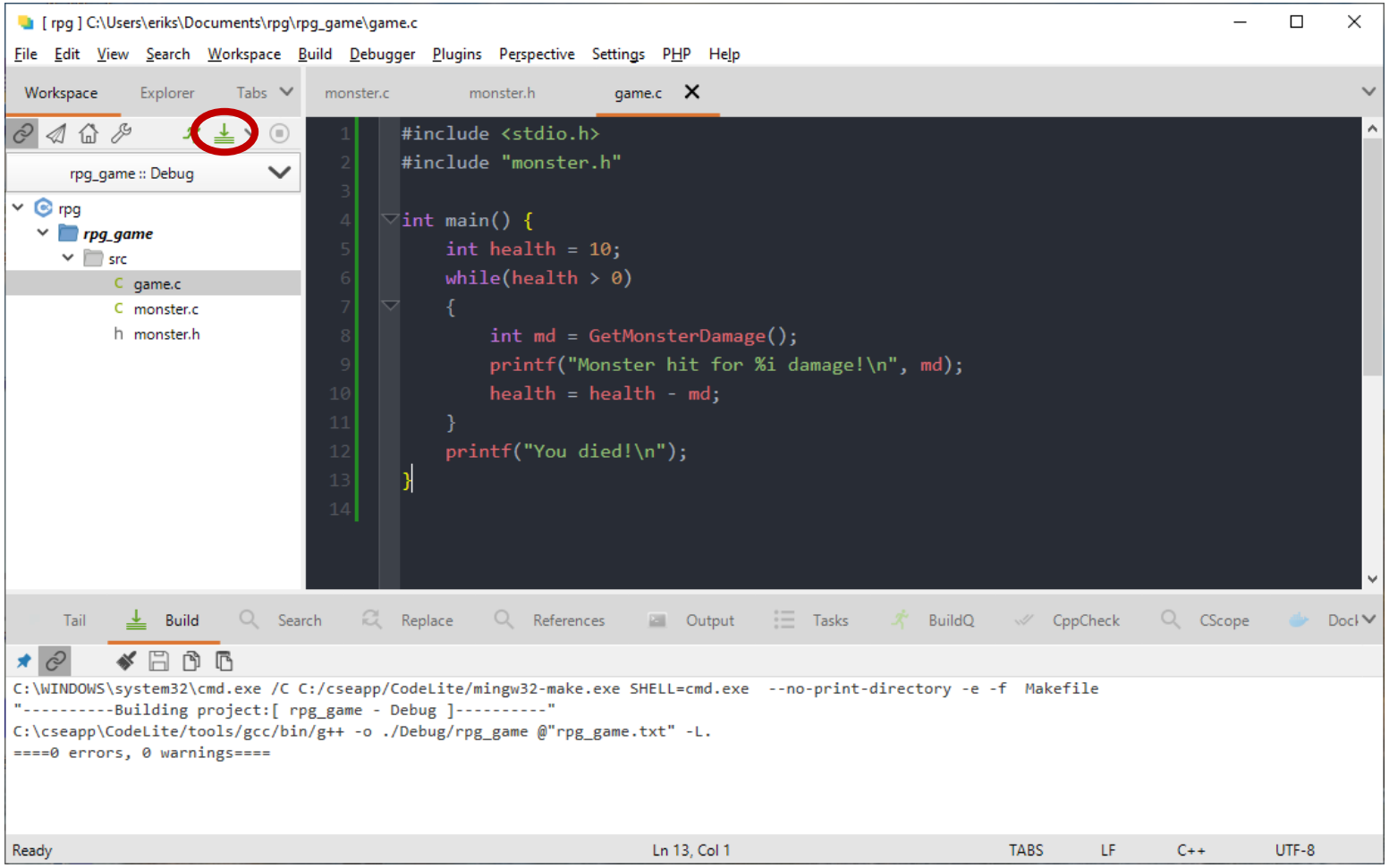
När programmen blir stora vill man dela upp i flera filer

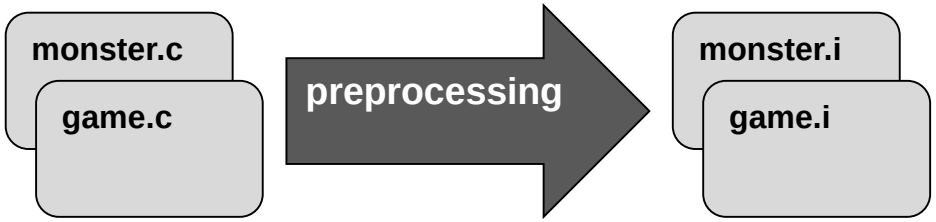
```c
// game.c
#include <stdio.h>
#include "monster.h"

int main() {
    int health = 10;
    while(health > 0)
    {
        int md = GetMonsterDamage();
        printf("Monster hit for %i damage!\n", md);
        health = health - md;
    }
    printf("You died!\n");
}
```

```c
// monster.h
#define MONSTER_STRENGTH 1
#define MONSTER_WEAPON 2

int GetMonsterDamage();
```

```c
// monster.c
#include "monster.h"
int GetMonsterDamage()
{
    return MONSTER_STRENGTH * MONSTER_WEAPON;
}
```

Vad händer när vi trycker på "Build"?





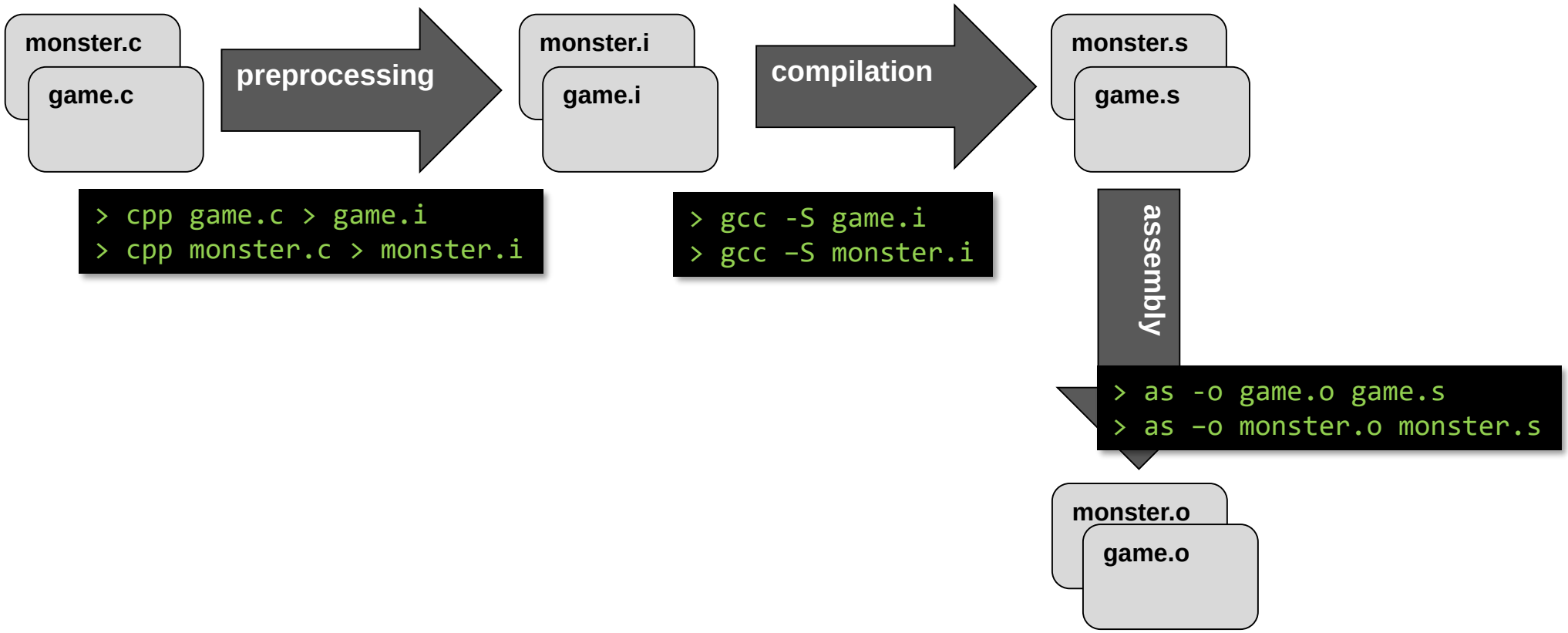

```
> cpp game.c > game.i
> cpp monster.c > monster.i
```

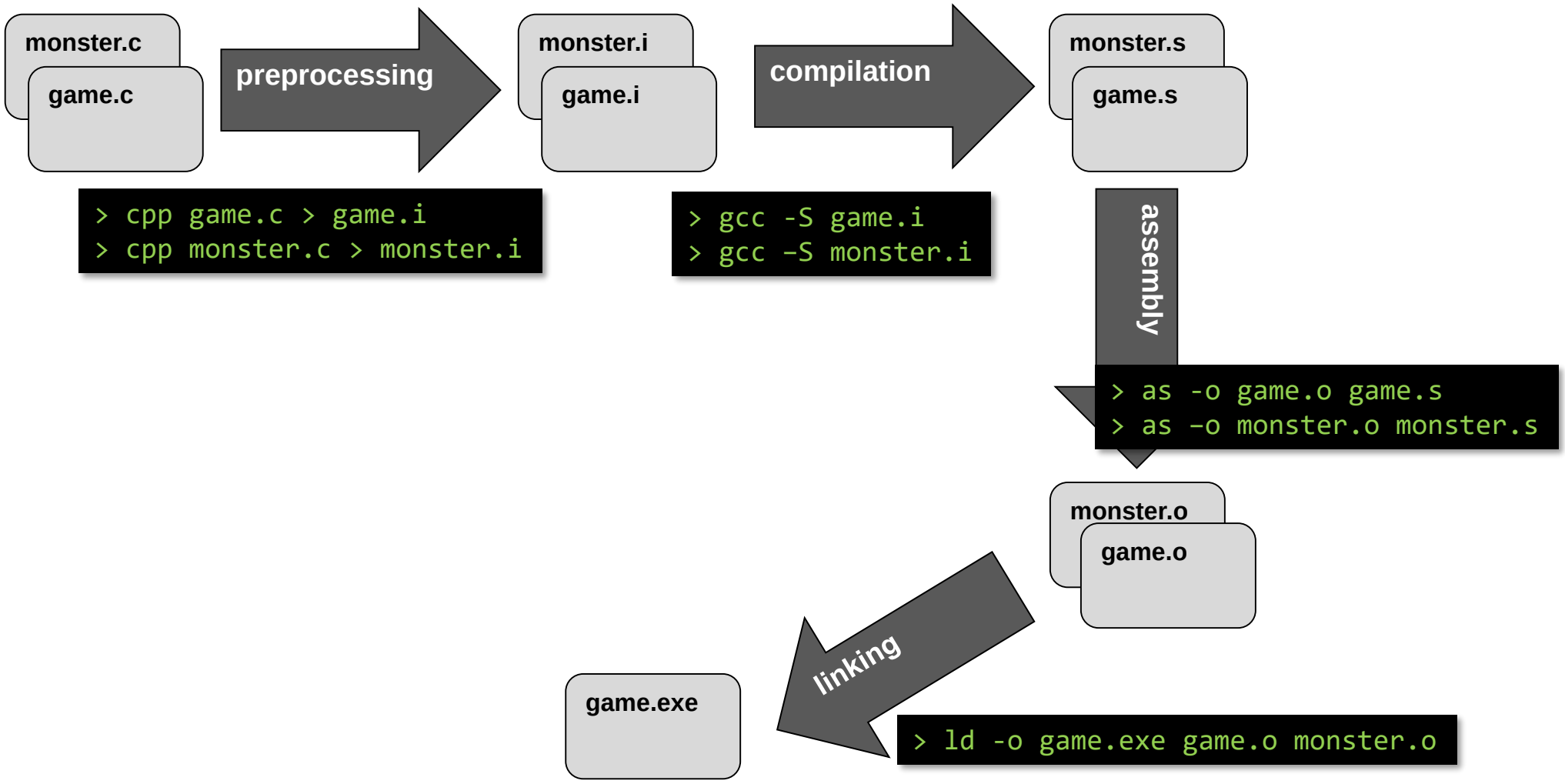
```h
// monster.h
#define MONSTER_STRENGTH 1
#define MONSTER_WEAPON 2

int GetMonsterDamage();
```

```c
// monster.c
#include "monster.h"
int GetMonsterDamage()
{
    return MONSTER_STRENGTH * MONSTER_WEAPON;
}
```


```
> cpp game.c > game.i
> cpp monster.c > monster.i
```

```
> gcc -S game.i
> gcc –S monster.i
```


monster.c
game.c
preprocessing
monster.i
game.i
compilation
monster.s
game.s
> cpp game.c > game.i
> cpp monster.c > monster.i
> gcc -S game.i
> gcc –S monster.i
assembly
> as -o game.o game.s
> as –o monster.o monster.s
monster.o
game.o


monster.c
game.c
preprocessing
monster.i
game.i
compilation
monster.s
game.s
> cpp game.c > game.i
> cpp monster.c > monster.i
> gcc -S game.i
> gcc –S monster.i
assembly
> as -o game.o game.s
> as –o monster.o monster.s
monster.o
game.o
linking
game.exe
> ld -o game.exe game.o monster.o

Arithmetic Operators

Basic assignment		a **=** b
Addition		a **+** b
Subtraction		a **-** b
Unary plus (integer promotion)		**+**a
Unary minus (additive inverse)		**-**a
Multiplication		a ***** b
Division		a **/** b
Modulo (integer remainder)		a **%** b
Increment	Prefix	**++**a
	Postfix	a**++**
Decrement	Prefix	**--**a
	Postfix	a**--**

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Arithmetic_operators

Comparison Operators

Equal to	a **==** b
Not equal to	a **!=** b
Greater than	a **>** b
Less than	a **<** b
Greater than or equal to	a **>=** b
Less than or equal to	a **<=** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Comparison_operators.2Frelational_operators

Logical Operators

Logical negation (NOT)	**!**a
Logical AND	a **&&** b
Logical OR	a **||** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Logical_operators

Compound Assignment Operators

Operator name	Syntax	Meaning
Addition assignment	a **+=** b	a **=** a **+** b
Subtraction assignment	a **-=** b	a **=** a **-** b
Multiplication assignment	a ***=** b	a **=** a ***** b
Division assignment	a **/=** b	a **=** a **/** b
Modulo assignment	a **%=** b	a **=** a **%** b
Bitwise AND assignment	a **&=** b	a **=** a **&** b
Bitwise OR assignment	a **|=** b	a **=** a **|** b
Bitwise XOR assignment	a **^=** b	a **=** a **^** b
Bitwise left shift assignment	a **<<=** b	a **=** a **<<** b
Bitwise right shift assignment	a **>>=** b	a **=** a **>>** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Compound_assignment_operators

Bitwise Operators

Bitwise NOT	**~**a
Bitwise AND	a **&** b
Bitwise OR	a **|** b
Bitwise XOR	a **^** b
Bitwise left shift	a **<<** b
Bitwise right shift	a **>>** b

http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Bitwise_operators

Bitwise operatorer, Exempel

```
int IsSetMSB(int x)
{
    int masked = x & 0x8000000;
    if(masked == 0) return 0;
    else return 1;
}
```

Bitwise operatorer, Exempel

```
int IsSetMSB(int x)
{
    int masked = x & 0x8000000;
    if(masked == 0) return 0;
    else
}
```

```
int SetMSB(int x)
{
    return x | 0x8000000;
}
```

Bitwise operatorer, Exempel

```
int IsSetMSB(int x)
{
    int masked = x
    if(masked == 0)
    else
}
```

```
int Se
{
    re
}
```

```
int CountOnes(int x)
{
    int num_ones = 0;
    for(int i=0; i<32; i++)
    {
        if((x & 0x1) == 0x1) num_ones += 1;
        x = x >> 1;
    }
    return num_ones;
}
```

Nästa C lektion

- Structs (komposit datatyp)
- Pekare
- Portadressering

- Gå igenom övningsexemplen på Canvas