C programming language

College of Saint Benedict & Saint John's University

origins

Dennis Ritchie in 2011 / CC BY 2.0

Brian Kernighan in 2012 / CC BY 2.0

hello, world

```c
/* file: helloworld.c */

#include <stdio.h>

int main() {
  printf("hello, world\n");
  return 0;
}
```

```
$ gcc -o helloworld helloworld.c
$ ./helloworld
hello, world
```

global variables

```c
// file: figure2-4.c
// Stan Warford
// A nonsense program to illustrate global variables

#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

```
$ gcc -o figure2-4 figure2-4.c
$ ./figure2-4
M 419
N
424
```

program breakdown

```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

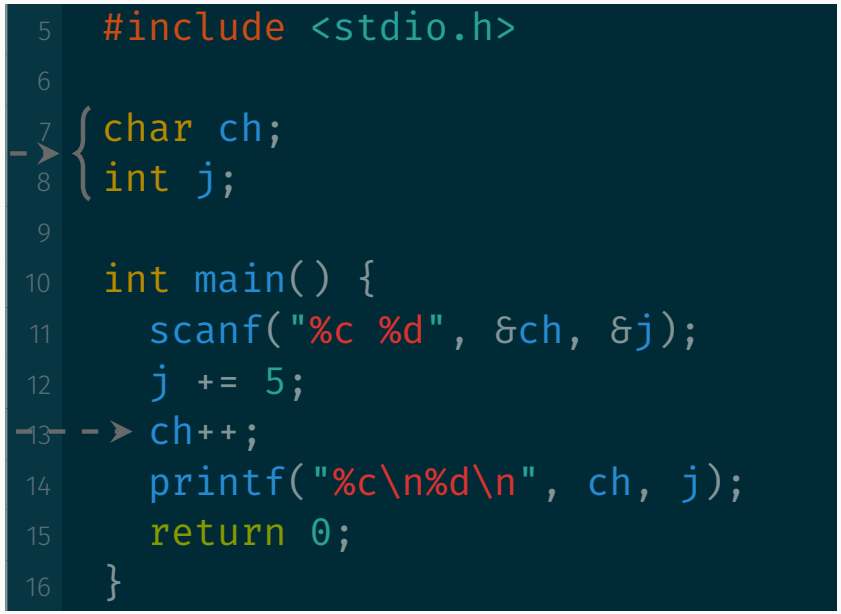
C programs ALWAYS start execution with the **main** function

returning from **main** ends the program

program breakdown

global variables are declared here — outside of any function

characters in C are treated internally like signed integers

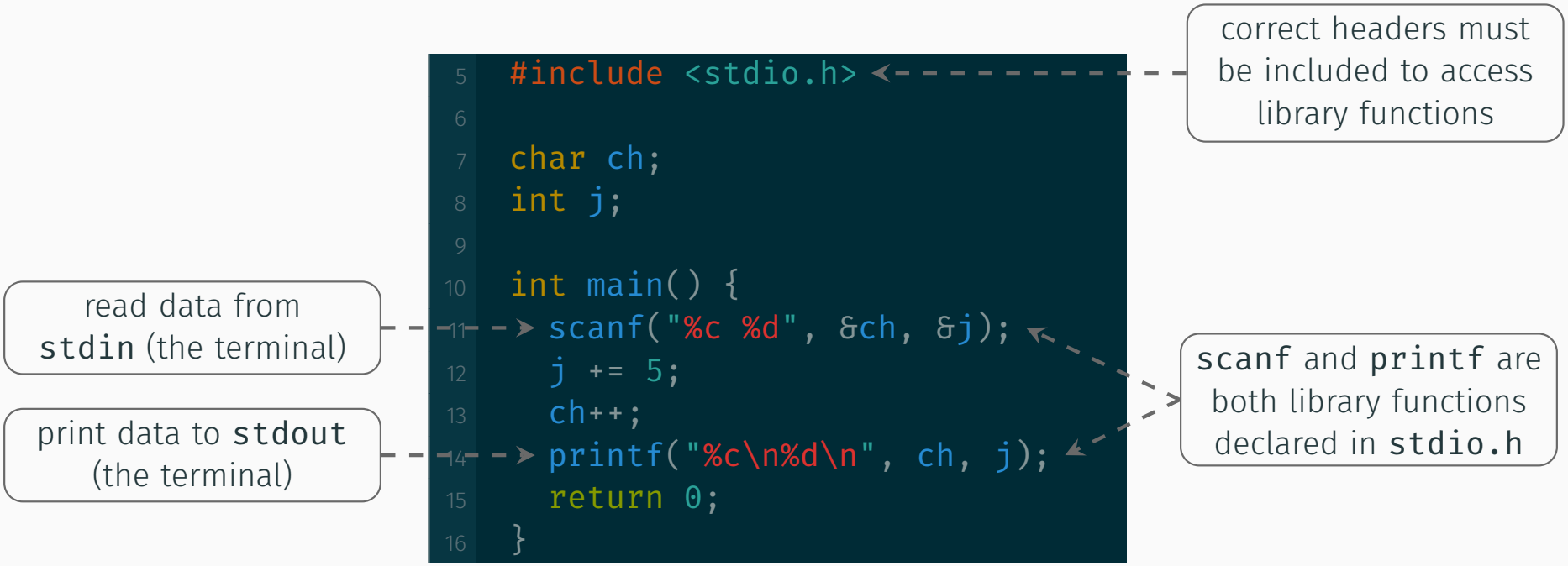


```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

program breakdown



program breakdown

```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

`&` is the address of operator — `scanf` expects the address of the variables where the data will be stored

conditions

```
if (<cond>) {
  /* ... */
}
else (<cond>) {
  /* ... */
}
else {
  /* ... */
}
```

conditions

```
1 if (x) {
2   /* ??? */
3 }
4 if (x-y) {
5   /* ??? */
6 }
7 if (x=y) {
8   /* ??? */
9 }
```

- under what conditions will each of the above be executed?

switch

```
switch (<expr>) {
  case <const>:
  /* ... */

  case <const>: /* fall-through */
  /* ... */
  break;

  default:
  /* ... */
  break;
}
```

loops

```
for (<init>; <cond>; <incr>) {
  /* ... */
}

while (<cond>)
  /* ... */
}

do {
  /* ... */
} while (<cond>);
```

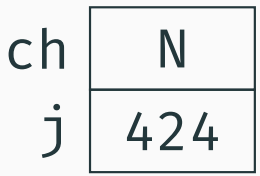
memory model — part i

global variables

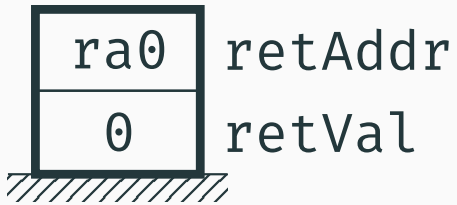
declared outside of any function and remain in place throughout the execution of the entire program. they are stored at a fixed location in memory.

local variables

declared within a function and come into existence when the function is called and cease to exist when the function terminates. they are stored on the run-time stack.



(a) Fixed location.



(b) Run-time stack.

run-time stack a.k.a. “the stack”

run-time stack

stores information about the active functions of a C program, including:

- return value,
- parameters,
- return address, and
- local variables

in that order.

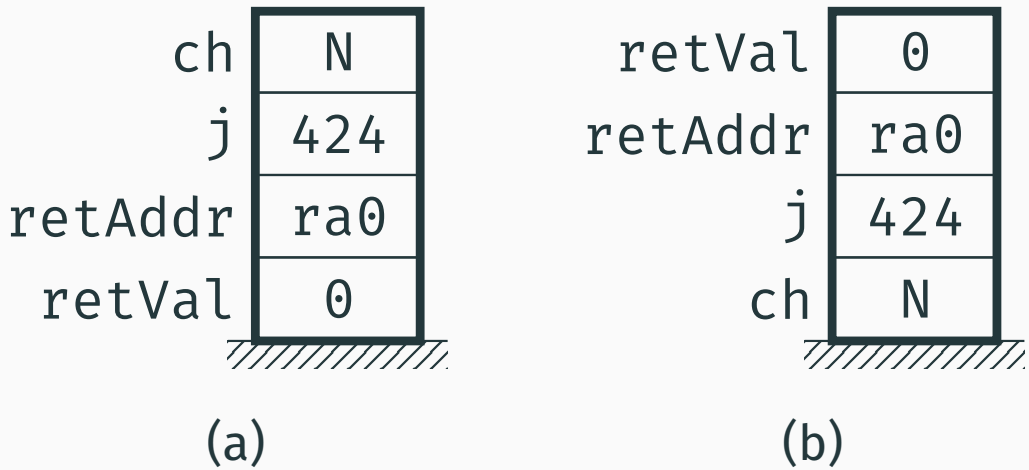
run-time stack a.k.a. “the stack”

run-time stack

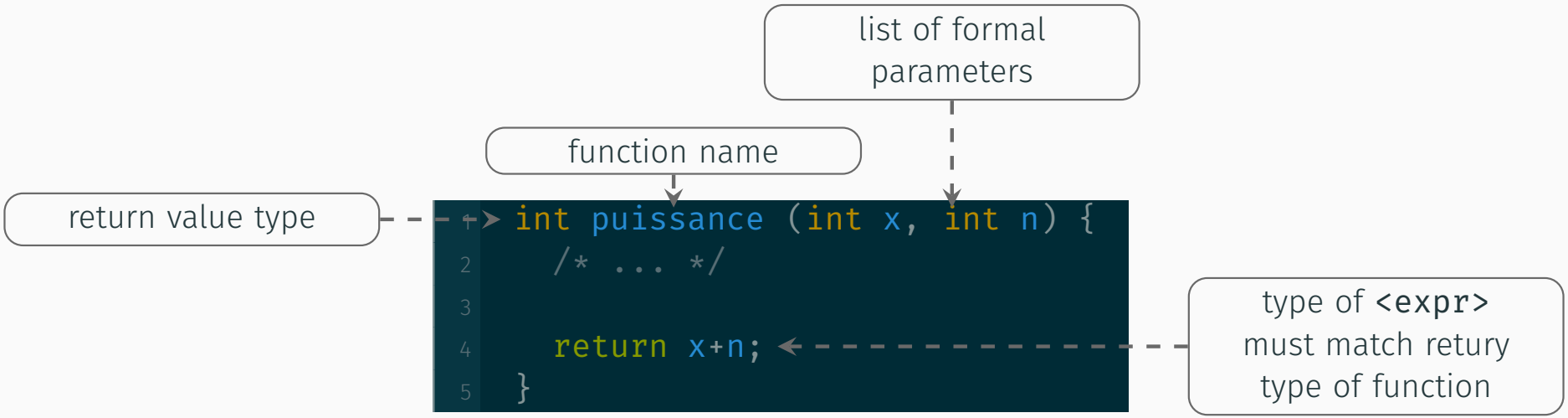
stores information about the active functions of a C program, including:

- return value,
- parameters,
- return address, and
- local variables

in that order.



functions


list of formal parameters
function name
return value type
1 int puissance (int x, int n) {
2 /* ... */
3
4 return x+n;
5 }
type of <expr> must match retury type of function

comparison

Java	C
object-oriented	procedural
interpreted	compiled
`String`	`char` array
condition (`boolean`)	condition (`int`)
garbage-collected	no memory management
references	pointers
exceptions	error codes