中南大學
CENTRAL SOUTH UNIVERSITY

第十讲

程　序　结　构

10.1 局部变量

- 局部变量：函数体内声明的变量

```
int sum_digits(int n){
  int sum = 0;    /* local variable */

  while (n > 0) {
    sum += n % 10;
    n /= 10;
  }
  return sum;
}
```

sum的作用域

函数被调用时分配

函数返回时回收

作用域在所属函数内部

10.1 局部变量

- 静态局部变量：在整个程序执行期间都会保留变量的值
- 函数返回后，值依然保留，但是对其他函数不可见

```c
#include <stdio.h>
void f(){
    int i=0;
    i++;
    printf("%d\n",i);
}
int main(){
    f();
    f();
    f();

    return 0;

}
```

```
1
1
1
```

```c
#include <stdio.h>
void f(){
    static int i=0;
    i++;
    printf("%d\n",i);
}
int main(){
    f();
    f();
    f();

    return 0;

}
```

```
1
2
3
```

10.2 外部变量

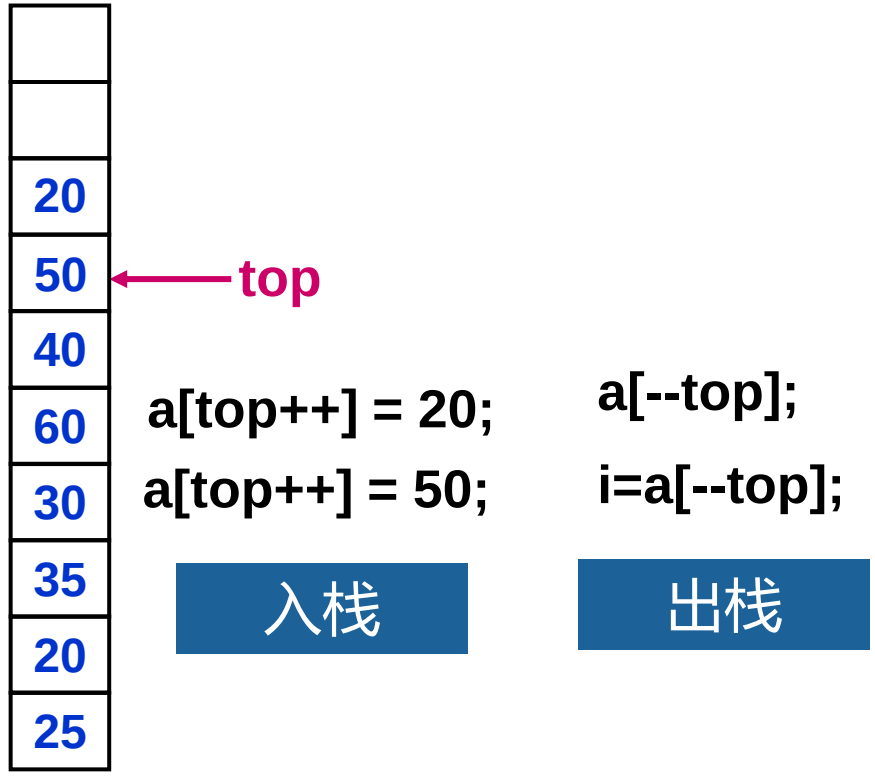
- 外部变量（全局变量）
  - 同static，程序执行期间都会保留变量的值，可以被文件类所有函数访问（并修改）
  - 文件作用域：从声明开始到文件的末尾，外部变量声明后的所有函数都能访问（并修改）

堆栈的实现

```
#define STACK_SIZE 10   /*堆栈容量*/
```

栈空 top == 0

栈满top == STACK_SIZE



10.2 外部变量

堆栈的编码实现（用外部变量）

```c
#include <stdbool.h>    /* C99 only */
#define STACK_SIZE 100
/* external variables */
int contents[STACK_SIZE];
int top = 0;
void make_empty(void){
    top = 0;
}
bool is_empty(void){
    return top == 0;
}
bool is_full(void){
    return top == STACK_SIZE;
}
```

```c
void push(int i){
    if (is_full()
        printf("The stack is full!\n");
    else
        contents[top++] = i;
}
int pop(void){
    if (is_empty())
        printf("The stack is empty!\n");
    else
        return contents[--top];
}
```

10.2 外部变量

- 外部变量的利弊
  - 大多数情况下，通过形参传递值比共享变量更好

```
int i;
void print_one_row(void){
    for (i = 1; i <= 10; i++)
        printf("*");
}
void print_all_rows(void){
    for (i = 1; i <= 10; i++) {
        print_one_row();
        printf("\n");
    }
}
```

输出结果?

10.2 外部变量

- 示例，猜数

```
Guess the secret number between 1 and 100.

A new number has been chosen.
Enter guess: 50
Too low; try again.
Enter guess: 75
Too high; try again.
Enter guess: 63
Too low; try again.
Enter guess: 69
Too low; try again.
Enter guess: 72
Too high; try again.
Enter guess: 70
Too low; try again.
Enter guess: 71
You won in 7 guesses!

Play again? (Y/N)
```

10.2 外部变量

- 示例，猜数

```c
/* Asks user to guess a hidden number */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_NUMBER 100
/* external variable */
int secret_number;
/* prototypes */
void initialize_number_generator(void);
void choose_new_secret_number(void);
void read_guesses(void);
```

```c
int main(void)
{
  char command;
  printf("Guess the secret number \
          between 1 and %d.\n\n",MAX_NUMBER);
  initialize_number_generator();
  do {
    choose_new_secret_number();
    printf("A new number has been chosen.\n");
    read_guesses();
    printf("Play again? (Y/N) ");
    scanf(" %c", &command);
    printf("\n");
  } while (command == 'y' || command == 'Y');
  return 0;
}
```

10.2 外部变量

- 示例，猜数

```
void initialize_number_generator(void)
{
  srand((unsigned) time(NULL));
}
void choose_new_secret_number(void)
{
  secret_number = rand() % MAX_NUMBER + 1;
}
```

```
void read_guesses(void)
{
  int guess, num_guesses = 0;
  for (;;) {
    num_guesses++;
    printf("Enter guess: ");
    scanf("%d", &guess);
    if (guess == secret_number) {
      printf("You won in %d guesses!\n\n",
             num_guesses);
      return;
    } else if (guess < secret_number)
      printf("Too low; try again.\n");
    else
      printf("Too high; try again.\n");
  }
}
```

10.3 程序块

- 程序块：包含声明的符合语句

交换i和j的值

```
if (i > j) {
    int temp = i;
    i = j;
    j = temp;
}
```

临时变量

- 程序块中声明的变量作用域局限于程序块
- 避免与其他同名变量冲突

10.4 作用域

- 同名标识符新的声明临时隐藏旧声明

- 声明1：全局变量，文件作用域
- 声明2：形式参数，块作用域
- 声明3：局部变量，块作用域
- 声明3：临时变量，块作用域

```
int i ;                 /* Declaration 1 */

void f(int i )          /* Declaration 2 */
{
   i = 1;
}

void g(void)
{
   int i = 2;           /* Declaration 3 */

   if (i > 0) {
      int i ;           /* Declaration 4 */

      i = 3;
   }

   i = 4;
}

void h(void)
{
   i = 5;
}
```

10.5 构建C程序

- 预处理指令
- 类型定义
- 外部变量声明
- 函数原型
- 函数定义

- #include指令
- #define指令
- 类型定义
- 外部变量的声明
- 除main之外的函数的原型
- main函数的定义
- 其他函数的定义