知识点总结

·在C语言中，函数是一段具有特定功能的、可重复使用的代码块。它有助于将复杂的程序分解为较小的、易于管理的部分，提高代码的可读性、可维护性和可重用性。

·函数的定义：函数定义由函数头和函数体两部分组成。

语法格式：

```
// 函数头
返回值类型 函数名(参数类型1 参数名1, 参数类型2 参数名2, ...) {
    // 函数体
    语句;
    return 返回值;
}
```

其中，返回值类型确定了函数执行完毕后返回值的数据类型，即return语句中返回值的类型。若函数无需返回值，使用关键字void指定返回值类型。

·语法格式：

```
// 函数头
返回值类型 函数名(参数类型1 参数名1, 参数类型2 参数名2, ...) {
    // 函数体
    语句;
    return 返回值;
}
```

·函数名后的括号用于指定函数所需外部提供的参数，参数个数可为零个或多个。每个参数由数据类型与参数名构成，多个参数间以逗号分隔。若函数无需参数，括号为空。

·函数体包含函数具体执行的代码。

·return语句用于终止函数执行，并将返回值传递至调用函数处。对于没有返回值的函数，可以不使用return语句，或者使用return;来提前结束函数。

例1：定义一个用于计算两个整数的和的函数（需要返回值）。

```
// 返回值result的类型是int，返回值类型为int
// 函数名：add
// 需要两个int类型参数：a，b
int add(int a, int b) {
    // 计算a和b的和
    int result = a + b;
    // 返回计算结果
    return result;
}
```

例2：定义一个用于打印唠嗑开场白的函数（不需要返回值）。

```
// 不需要返回值，返回值类型为void
// 函数名：printf_greetings
// 不需要参数，函数名后面括号为空
void printf_greetings() {
    printf("好久不见！\n");
    printf("今天天气不错哈。\n");
    printf("你吃饭了吗？\n");
    // 没有返回值，可以省略return语句
}
```

·函数的声明：函数声明告诉编译器函数的名称、参数类型和返回值类型。

·函数声明和函数定义的区别：

函数声明只是告诉编译器函数的名字、参数类型和返回值类型，不包含函数的具体实现代码。

函数定义要实现函数的具体功能，包含函数所有的可执行代码。

·函数声明的语法格式：

1）基本形式（省略参数名）

返回值类型 函数名(参数类型1, 参数类型2, ...);

2）带参数名的形式

返回值类型 函数名(参数类型1 参数名1, 参数类型2 参数名2, ...);

例：

1）基本形式：`int add(int, int);`

2）带参数名的形式：`int add(int a, int b);`

· 注意：

1、函数需要先声明再调用，但如果函数定义在调用之前，可以省略函数声明。

2、函数不能定义在main函数里，因为C语言里规定不允许函数嵌套定义，即不能在一个函数的函数体里，再定义另一个函数。

·调用函数的语法格式：

有参调用：函数名(参数1, 参数2, ...);

无参调用：函数名();

例2：调用printf_greetings函数（无参调用）。

```c
#include <stdio.h>
void printf_greetings() {
    printf("好久不见！\n");
    printf("今天天气不错哈。\n");
    printf("你吃饭了吗？\n");
}

int main() {
    // 调用printf_greetings函数
    printf_greetings();
}
```

·注意：调用函数时，提供的数据要和函数定义里的参数在数量、类型和顺序上都要保持一致。

例：

```
#include <stdio.h>

void func(int a, double b) { // 函数定义：参数类型依次为int, double
    ...
}
int main() {
    func(10, 3.14);  // 正确调用：参数数量、类型、顺序完全匹配

    // 错误调用
    func(10); // 参数过少（缺少第二个参数）
    func(10, 3.14, 'A'); // 参数过多（多了一个额外的参数）
    func("10", 3.14);   // 第一个参数类型完全错误
    func(3.14, 10); // 参数顺序错误导致类型不匹配
}
```

·补充：数组作为函数参数时，会被隐式转换为指向首元素的指针。因此，在函数声明和定义时，用指针类型表示数组参数的效果是一样的。

例：定义一个参数包含一维数组的函数。

```
//以下两种写法行为相同
//写成数组形式
void iterate_array(int arr[], int len) {
    ...
}
//写成指针形式
void iterate_array(int* arr, int len) {
    ...
}
```

函数声明时，参数的类型必须和函数定义时的参数类型严格匹配，但声明时，参数名可以省略不写
数组作为函数参数时，会被隐式转换为指向首元素的指针
在函数声明和定义时，用指针类型表示数组参数的效果是一样的

题目1、若有以下函数首部

```
int fun(double x[10], int* n)
```

则下面针对此函数的函数声明语句中正确的是（A）

A. int fun(double*, int*);

B. int fun(double, int);

C. int fun(double* x, int n);

D. int fun(double x, int* n);

char类型只能储存单个字符

题目1、设有函数定义：

```
void sub(int k,char ch){…}
```

则以下对函数sub的调用语句中，正确的是（A）

A．sub(1, 'a');

B．sub(2, 'aa');

C．h = sub(3, 'a');

D．sub(4, "a");

题目3、有以下程序:

```
#include <stdio.h>
int f(int x);
main()
{
    int a, b = 0;
    for(a = 0; a < 3; a++)
    {
        b = b + f(a);
        putchar('A' + b);
    }
}
int f(int x)
{
    return x * x + 1;
}
```

程序运行后的输出结果是（B）

A. ABE

B. BDI

C. BCF

D. BCD