BIT-1-初识C语言

> 基本了解C语言的基础知识，对C语言有一个大概的认识。
>
> 每个知识点就是简单认识，不做详细讲解，后期课程都会细讲。

本章重点：

- 什么是C语言
- 第一个C语言程序
- 数据类型
- 变量、常量
- 字符串+转义字符+注释
- 选择语句
- 循环语句
- 函数
- 数组
- 操作符
- 常见关键字
- define 定义常量和宏
- 指针
- 结构体

正文开始@比特就业课

1. 什么是C语言?

> C语言是一门通用计算机编程语言，广泛应用于底层开发。C语言的设计目标是提供一种能以简易的方式编译、处理低级存储器、产
>
> 生少量的机器码以及不需要任何运行环境支持便能运行的编程语言。

> 尽管C语言提供了许多低级处理的功能，但仍然保持着良好跨平台的特性，以一个标准规格写出的C语言程序可在许多电脑平台上进
>
> 行编译，甚至包含一些嵌入式处理器（单片机或称MCU）以及超级电脑等作业平台。

> 二十世纪八十年代，为了避免各开发厂商用的C语言语法产生差异，由美国国家标准局为C语言制定了一套完整的美国国家标准语
>
> 法，称为ANSI C，作为C语言最初的标准。 [1] 目前2011年12月8日，国际标准化组织（ISO）和国际电工委员会（IEC）发布的**C11**
>
> **标准**是C语言的第三个官方标准，也是C语言的最新标准，该标准更好的支持了汉字函数名和汉字标识符，一定程度上实现了汉
>
> 字编程。

> C语言是一门面向过程的计算机编程语言，与C++，Java等面向对象的编程语言有所不同。

> 其编译器主要有Clang、**GCC**、WIN-TC、SUBLIME、**MSVC**、Turbo C等。

2. 第一个C语言程序

```c
#include <stdio.h>

int main()
{
    printf("hello bit\n");
    printf("he he\n");
    return 0;
}
//解释:
//main函数是程序的入口
//一个工程中main函数有且仅有一个
```

3. 数据类型

```c
char        //字符数据类型
short       //短整型
int         //整形
long        //长整型
long long   //更长的整形
float       //单精度浮点数
double      //双精度浮点数
//C语言有没有字符串类型?
```

- 为什么出现这么的类型?
- 每种类型的大小是多少?

```c
#include <stdio.h>
int main()
{
    printf("%d\n", sizeof(char));
    printf("%d\n", sizeof(short));
    printf("%d\n", sizeof(int));
    printf("%d\n", sizeof(long));
    printf("%d\n", sizeof(long long));
    printf("%d\n", sizeof(float));
    printf("%d\n", sizeof(double));
    printf("%d\n", sizeof(long double));
    return 0;
}
```

注意:存在这么多的类型,其实是为了更加丰富的表达生活中的各种值。

类型的使用:

```c
char ch = 'w';
int weight = 120;
int salary = 20000;
```

3. 变量、常量

生活中的有些值是不变的(比如:圆周率,性别,身份证号码,血型等等)

有些值是可变的(比如:年龄,体重,薪资)。

不变的值，C语言中用**常量**的概念来表示，变得值C语言中用**变量**来表示。

3.1 定义变量的方法

```
int age = 150;
float weight = 45.5f;
char ch = 'w';
```

3.2 变量的分类

- 局部变量
- 全局变量

```
#include <stdio.h>

int global = 2019;//全局变量
int main()
{
    int local = 2018;//局部变量
    //下面定义的global会不会有问题?
    int global = 2020;//局部变量
    printf("global = %d\n", global);
    return 0;
}
```

总结:

> 上面的局部变量global变量的定义其实没有什么问题的!
>
> 当局部变量和全局变量同名的时候，局部变量优先使用。

3.3 变量的使用

```
#include <stdio.h>
int main()
{
    int num1 = 0;
    int num2 = 0;
    int sum = 0;
    printf("输入两个操作数:>");
    scanf("%d %d", &num1, &num2);
    sum = num1 + num2;
    printf("sum = %d\n", sum);
    return 0;
}
//这里介绍一下输入，输出语句
//scanf
//printf
```

3.4 变量的作用域和生命周期

作用域

> 作用域（scope）是程序设计概念，通常来说，一段程序代码中所用到的名字并不总是有效/可用的
>
> 而限定这个名字的可用性的代码范围就是这个名字的作用域。

1. 局部变量的作用域是变量所在的局部范围。
2. 全局变量的作用域是整个工程。

生命周期

> 变量的生命周期指的是变量的创建到变量的销毁之间的一个时间段

1. 局部变量的生命周期是：进入作用域生命周期开始，出作用域生命周期结束。
2. 全局变量的生命周期是：整个程序的生命周期。

3.5 常量

C语言中的常量和变量的定义的形式有所差异。

C语言中的常量分为以下以下几种：

- 字面常量
- `const` 修饰的常变量
- #define 定义的标识符常量
- 枚举常量

```c
#include <stdio.h>
//举例
enum Sex
{
    MALE,
    FEMALE,
    SECRET
};
//括号中的MALE,FEMALE,SECRET是枚举常量

int main()
{
    //字面常量演示
    3.14;//字面常量
    1000;//字面常量

    //const 修饰的常变量
    const float pai = 3.14f;    //这里的pai是const修饰的常变量
    pai = 5.14;//是不能直接修改的!

    //#define的标识符常量 演示
    #define MAX 100
    printf("max = %d\n", MAX);

    //枚举常量演示
    printf("%d\n", MALE);
    printf("%d\n", FEMALE);
    printf("%d\n", SECRET);
    //注：枚举常量的默认是从0开始，依次向下递增1的
```

```
    return 0;
}
```

注：

上面例子上的`pai`被称为`const`修饰的常变量，`const`修饰的常变量在C语言中只是在语法层面限制了变量`pai`不能直接被改变，但是`pai`本质上还是一个变量的，所以叫常变量。

4. 字符串+转义字符+注释

4.1 字符串

```
"hello bit.\n"
```

这种由双引号（Double Quote）引起来的一串字符称为字符串字面值（String Literal），或者简称字符串。

注：字符串的结束标志是一个`\0`的转义字符。在计算字符串长度的时候`\0`是结束标志，不算作字符串内容。

```
#include <stdio.h>
//下面代码，打印结果是什么？为什么？（突出'\0'的重要性）
int main()
{
    char arr1[] = "bit";
    char arr2[] = {'b', 'i', 't'};
    char arr3[] = {'b', 'i', 't', '\0'};
    printf("%s\n", arr1);
    printf("%s\n", arr2);
    printf("%s\n", arr3);
    return 0;
}
```

4.2 转义字符

加入我们要在屏幕上打印一个目录：`c:\code\test.c`

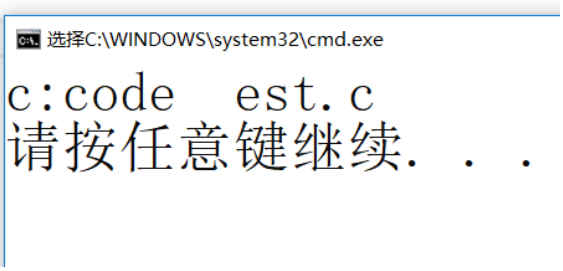
我们该如何写代码？

```
#include <stdio.h>

int main()
{
    printf("c:\code\test.c\n");
    return 0;
}
```

实际上程序运行的结果是这样的：

```c
#include <stdio.h>
int main()
{
    printf("c:\code\test.c\n");
    return 0;
}
```



这里就不得不提一下转义字符了。转义字符顾名思义就是转变意思。

下面看一些转义字符。

转义字符	释义
\?	在书写连续多个问号时使用，防止他们被解析成三字母词
\'	用于表示字符常量'
\“	用于表示一个字符串内部的双引号
\\	用于表示一个反斜杠，防止它被解释为一个转义序列符。
\a	警告字符，蜂鸣
\b	退格符
\f	进纸符
\n	换行
\r	回车
\t	水平制表符
\v	垂直制表符
\ddd	ddd表示1~3个八进制的数字。 如： \130 X
\xdd	dd表示2个十六进制数字。 如： \x30 0

```c
#include <stdio.h>
int main()
{
    //问题1：在屏幕上打印一个单引号'，怎么做？
    //问题2：在屏幕上打印一个字符串，字符串的内容是一个双引号“，怎么做？
    printf("%c\n", '\'');
    printf("%s\n", "\"");
    return 0;
}
```

笔试题：

```c
//程序输出什么？
#include <stdio.h>

int main()
{
    printf("%d\n", strlen("abcdef"));
    // \62被解析成一个转义字符
    printf("%d\n", strlen("c:\test\628\test.c"));
    return 0;
}
```

5. 注释

1. 代码中有不需要的代码可以直接删除，也可以注释掉
2. 代码中有些代码比较难懂，可以加一下注释文字

比如：

```c
#include <stdio.h>

int Add(int x, int y)
{
    return x+y;
}
/*C语言风格注释
int Sub(int x, int y)
{
    return x-y;
}
*/
int main()
{
    //C++注释风格
    //int a = 10;
    //调用Add函数，完成加法
    printf("%d\n", Add(1, 2));
    return 0;
}
```

注释有两种风格：

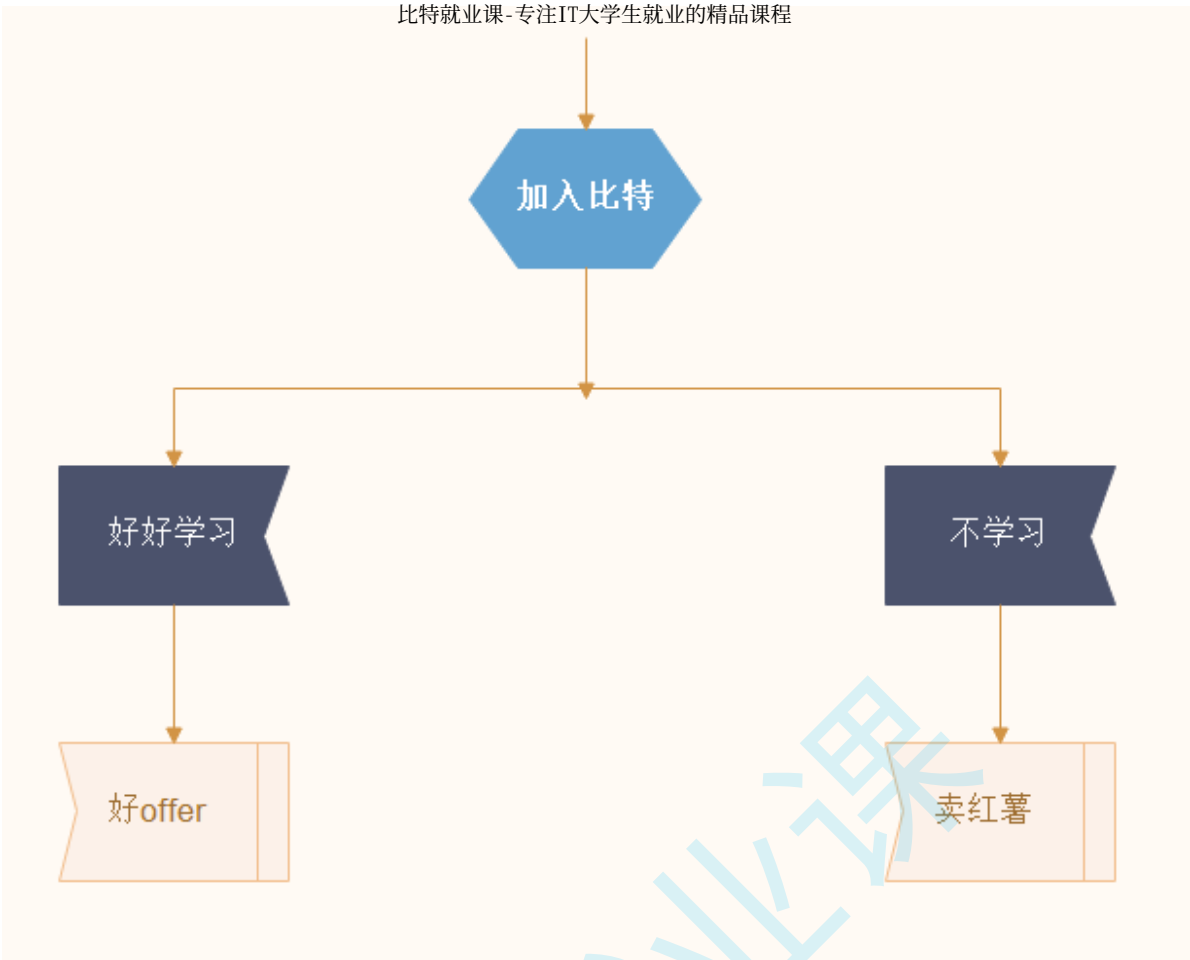
- C语言风格的注释 `/*xxxxxx*/`
  - 缺陷：不能嵌套注释
- C++风格的注释 `//xxxxxxxx`
  - 可以注释一行也可以注释多行

6. 选择语句

如果你好好学习，校招时拿一个好offer，走上人生巅峰。

如果你不学习，毕业等于失业，回家卖红薯。

这就是选择！



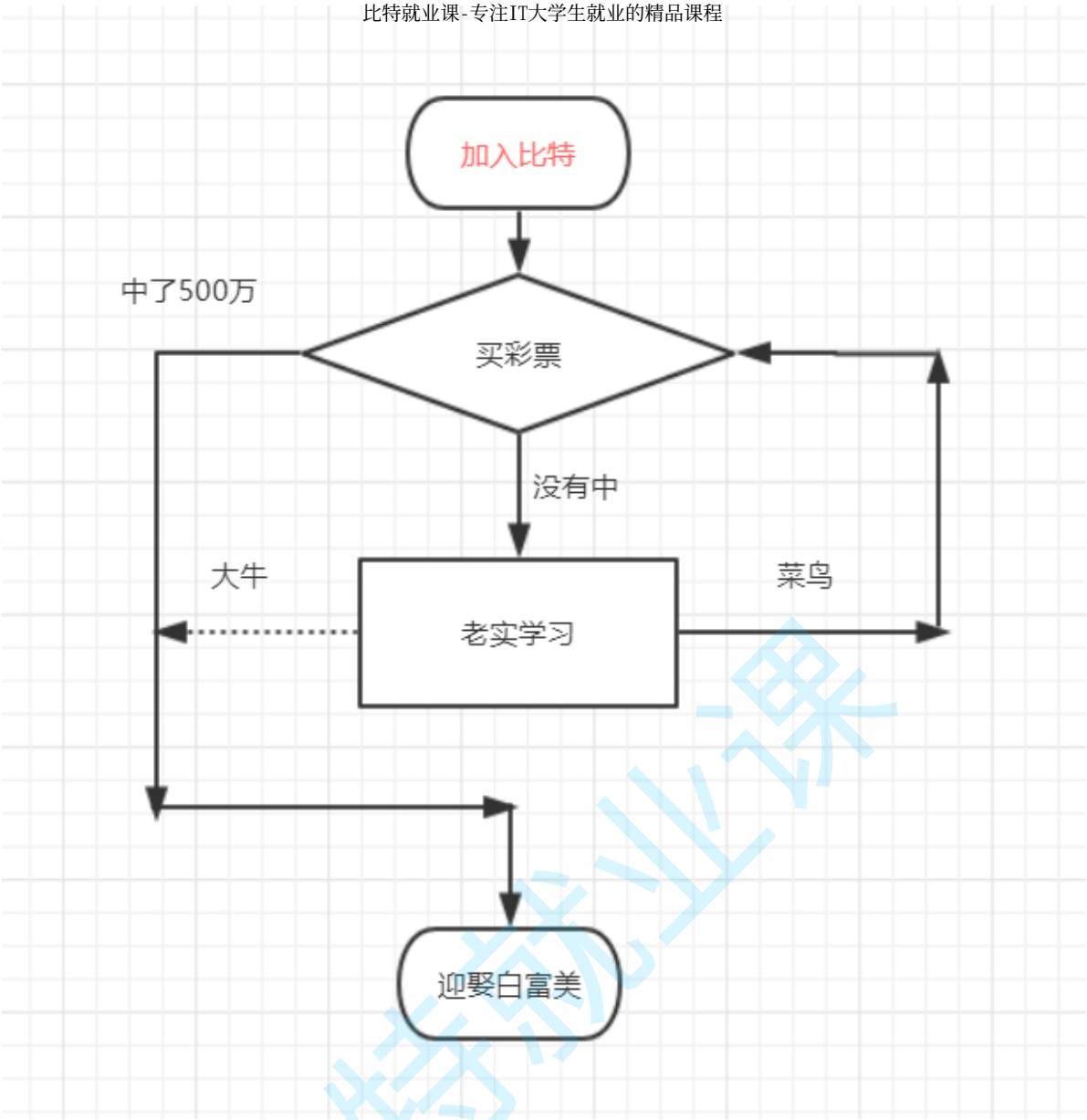

```c
#include <stdio.h>

int main()
{
    int coding = 0;
    printf("你会去敲代码吗？（选择1 or 0）:>");
    scanf("%d", &coding);
    if(coding == 1)
    {
        prinf("坚持，你会有好offer\n");
    }
    else
    {
        printf("放弃，回家卖红薯\n");
    }
    return 0;
}
```

7. 循环语句

有些事必须一直做，比如我日复一日的讲课，比如大家，日复一日的学习。

还比如：



C语言中如何实现循环呢?

- while语句-讲解
- for语句（后期讲）
- do ... while语句（后期讲）

```c
//while循环的实例
#include <stdio.h>

int main()
{
    printf("加入比特\n");
    int line = 0;
    while(line<=20000)
    {
        line++;
        printf("我要继续努力敲代码\n");
    }
    if(line>20000)
        printf("好offer\n");
    return 0;
}
```

8. 函数

```c
#include <stdio.h>

int main()
{
    int num1 = 0;
    int num2 = 0;
    int sum = 0;
    printf("输入两个操作数:>");
    scanf("%d %d", &num1, &num2);
    sum = num1 + num2;
    printf("sum = %d\n", sum);
    return 0;
}
```

上述代码，写成函数如下：

```c
#include <stdio.h>

int Add(int x, int y)
{
    int z = x+y;
    return z;
}

int main()
{
    int num1 = 0;
    int num2 = 0;
    int sum = 0;
    printf("输入两个操作数:>");
    scanf("%d %d", &num1, &num2);
    sum = Add(num1, num2);
    printf("sum = %d\n", sum);
    return 0;
}
```

函数的特点就是简化代码，代码复用。

9. 数组

要存储1-10的数字，怎么存储?

C语言中给了数组的定义：一组相同类型元素的集合

9.1 数组定义

```c
int arr[10] = {1,2,3,4,5,6,7,8,9,10};//定义一个整形数组，最多放10个元素
```

9.2 数组的下标

C语言规定：数组的每个元素都有一个下标，下标是从0开始的。

数组可以通过下标来访问的。

比如：

```
int arr[10] = {0};
//如果数组10个元素，下标的范围是0-9
```

int arr[10]	0	0	0	0	0	0	0	0	0	0
下标	0	1	2	3	4	5	6	7	8	9

9.3 数组的使用

```
#include <stdio.h>

int main()
{
    int i = 0;
    int arr[10] = {1,2,3,4,5,6,7,8,9,10};
    for(i=0; i<10; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

10. 操作符

简单介绍为主，后面课件重点讲。

> 算术操作符

```
+   -   *   /   %
```

> 移位操作符

```
>>  <<
```

> 位操作符

```
&   ^   |
```

赋值操作符

```
=   +=   -=   *=   /=   &=   ^=   |=   >>=   <<=
```

单目操作符

```
!           逻辑反操作
-           负值
+           正值
&           取地址
sizeof      操作数的类型长度（以字节为单位）
~           对一个数的二进制按位取反
--          前置、后置--
++          前置、后置++
*           间接访问操作符(解引用操作符)
(类型)      强制类型转换
```

关系操作符

```
>
>=
<
<=
!=      用于测试“不相等”
==      用于测试“相等”
```

逻辑操作符

```
&&          逻辑与
||          逻辑或
```

条件操作符

```
exp1 ? exp2 : exp3
```

逗号表达式

```
exp1, exp2, exp3, …expN
```

下标引用、函数调用和结构成员

```
[]  ()  .  ->
```

11. 常见关键字

```
auto  break  case  char  const  continue  default  do  double else  enum
extern float  for  goto  if  int  long  register  return  short  signed
 sizeof  static struct  switch  typedef union  unsigned  void  volatile  while
```

注：关键字，先介绍下面几个，后期遇到讲解。

11.1 关键字 typedef

> typedef 顾名思义是类型定义，这里应该理解为类型重命名。

比如：

```
//将unsigned int 重命名为uint_32, 所以uint_32也是一个类型名
typedef unsigned int uint_32;

int main()
{
    //观察num1和num2,这两个变量的类型是一样的
    unsigned int num1 = 0;
    uint_32 num2 = 0;
    return 0;
}
```

11.2 关键字static

> 在C语言中：
>
> static是用来修饰变量和函数的
>
> 1. 修饰局部变量-称为静态局部变量
> 2. 修饰全局变量-称为静态全局变量
> 3. 修饰函数-称为静态函数

11.2.1 修饰局部变量

```
//代码1
#include <stdio.h>
void test()
{
    int i = 0;
    i++;
    printf("%d ", i);
}
int main()
{
    int i = 0;
    for(i=0; i<10; i++)
    {
        test();
    }
    return 0;
}
//代码2
#include <stdio.h>
void test()
{
    //static修饰局部变量
    static int i = 0;
    i++;
    printf("%d ", i);
}
int main()
{
```

```
    int i = 0;
    for(i=0; i<10; i++)
    {
        test();
    }
    return 0;
}
```

对比代码1和代码2的效果理解static修饰局部变量的意义。

结论：

> static修饰局部变量改变了变量的生命周期
>
> 让静态局部变量出了作用域依然存在，到程序结束，生命周期才结束。

11.2.2 修饰全局变量

```
//代码1
//add.c
int g_val = 2018;
//test.c
int main()
{
    printf("%d\n", g_val);
    return 0;
}

//代码2
//add.c
static int g_val = 2018;
//test.c
int main()
{
    printf("%d\n", g_val);
    return 0;
}
```

代码1正常，代码2在编译的时候会出现连接性错误。

结论：

> 一个全局变量被static修饰，使得这个全局变量只能在本源文件内使用，不能在其他源文件内使用。

11.2.3 修饰函数

```
//代码1
//add.c
int Add(int x, int y)
{
    return c+y;
}
//test.c
int main()
{
    printf("%d\n", Add(2, 3));
    return 0;
```

```
}

//代码2
//add.c
static int Add(int x, int y)
{
    return c+y;
}
//test.c
int main()
{
    printf("%d\n", Add(2, 3));
    return 0;
}
```

代码1正常，代码2在编译的时候会出现连接性错误.

结论：

> 一个函数被static修饰，使得这个函数只能在本源文件内使用，不能在其他源文件内使用。

剩余关键字后续课程中陆续会讲解。

12. #define 定义常量和宏

```
//define定义标识符常量
#define MAX 1000

//define定义宏
#define ADD(x, y) ((x)+(y))

#include <stdio.h>

int main()
{
    int sum = ADD(2, 3);
    printf("sum = %d\n", sum);

    sum = 10*ADD(2, 3);
    printf("sum = %d\n", sum);

    return 0;
}
```

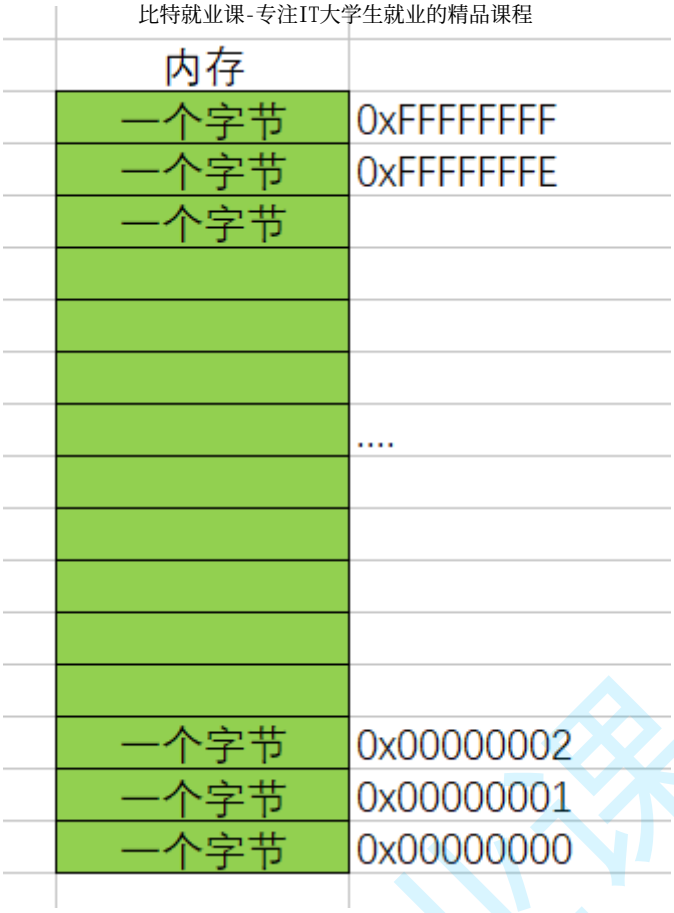
13. 指针

13.1 内存

内存是电脑上特别重要的存储器，计算机中程序的运行都是在内存中进行的 。

所以为了有效的使用内存，就把内存划分成一个个小的内存单元，每个内存单元的大小是**1个字节。**

为了能够有效的访问到内存的每个单元，就给内存单元进行了编号，这些编号被称为该**内存单元的地址。**



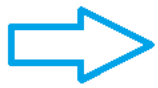
变量是创建内存中的（在内存中分配空间的），每个内存单元都有地址，所以变量也是有地址的。

取出变量地址如下：

```c
#include <stdio.h>

int main()
{
    int num = 10;
    &num;//取出num的地址
    //注：这里num的4个字节，每个字节都有地址，取出的是第一个字节的地址（较小的地址）
    printf("%p\n", &num);//打印地址，%p是以地址的形式打印
    return 0;
}
```

```c
int main()
{
    int num = 10;
    &num;//取出num的地址
    printf("%p\n", &num);
    return 0;
}
```



那地址如何存储，需要定义指针变量。

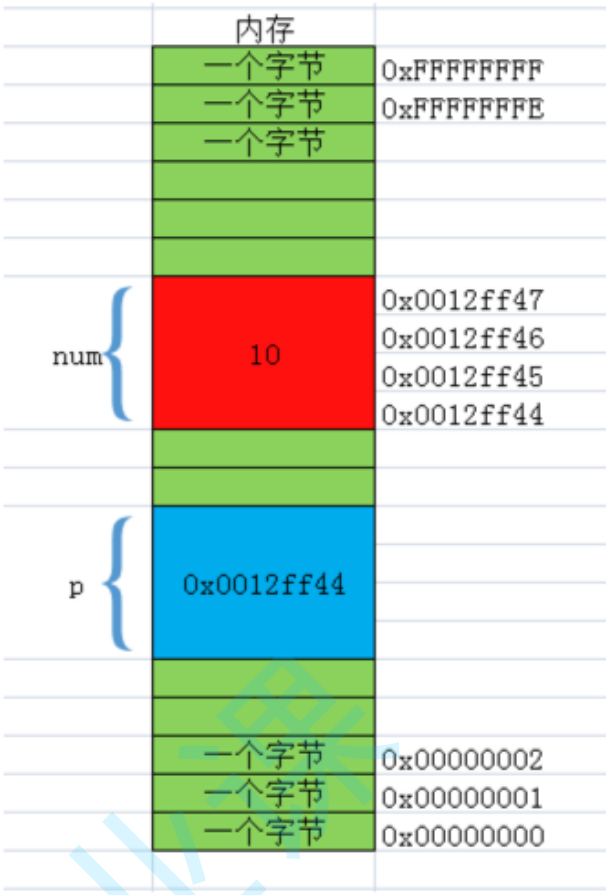
```c
int num = 10;
int *p;//p为一个整形指针变量
p = &num;
```

指针的使用实例：

```c
#include <stdio.h>

int main()
{
    int num = 10;
    int *p = &num;
    *p = 20;
    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int num = 10;
    int *p = &num;
    *p = 20;
    return 0;
}
```



以整形指针举例，可以推广到其他类型，如：

```c
#include <stdio.h>

int main()
{
    char ch = 'w';
    char* pc = &ch;
    *pc = 'q';
    printf("%c\n", ch);
    return 0;
}
```

13.2 指针变量的大小

```c
#include <stdio.h>
//指针变量的大小取决于地址的大小
//32位平台下地址是32个bit位（即4个字节）
//64位平台下地址是64个bit位（即8个字节）

int main()
{
    printf("%d\n", sizeof(char *));
    printf("%d\n", sizeof(short *));
    printf("%d\n", sizeof(int *));
    printf("%d\n", sizeof(double *));
    return 0;
}
```

结论：指针大小在32位平台是4个字节，64位平台是8个字节。

14. 结构体

结构体是C语言中特别重要的知识点，结构体使得C语言有能力描述复杂类型。

比如描述学生，学生包含：`名字+年龄+性别+学号`这几项信息。

这里只能使用结构体来描述了。

例如：

```
struct Stu
{
    char name[20];//名字
    int age;      //年龄
    char sex[5];  //性别
    char id[15];  //学号
};
```

结构体的初始化：

```
//打印结构体信息
struct Stu s = {"张三",  20,  "男",  "20180101"};

//.为结构成员访问操作符
printf("name = %s age = %d sex = %s id = %s\n", s.name, s.age, s.sex, s.id);
//->操作符
struct Stu *ps = &s;
printf("name = %s age = %d sex = %s id = %s\n", ps->name, ps->age, ps->sex, ps-
>id);
```

本章完。

限时福利

原价 99 元的鹏哥 C 语言集训营**（含 3 个月 1v1 答疑）**

限时抢购仅需 19.9 元

每天仅限前 100 名！！！

限时限额抢购鹏哥 C 语言集训营课程 + 3 个月 1V1 答疑服务

活动长期有效，但每天仅限前 100 名，扫描下方二维码立即快速抢购