

1.格式

```
165  格式语法:
166  变量名 = 数据;
167
168  int num;
169  num = 100;
```

2.赋值符号

171 1)。这个等号和你数学中的等号是完全不同的。
172 叫做赋值符号。
173 赋值符号的作用：将右边的数据 存储到 左边的变量所代表的空间中去。

3.int型变量赋值注意点

1>超过范围--报错

```
179 int num;
180     num = 199;
181
182
183
184
185
186
187
188
189
190
191
192 */
193 #include <stdio.h>
194
195 int main(int argc, const char * argv[])
196 {
197
198     int num;
199     num = 222121212121212121212121212121212123334546576;
200
201
```

Integer literal is too large to be represented in any integer type

2>赋值小数--不报错

```
199  */
200  #include <stdio.h>
201
202  int main(int argc, const char * argv[])
203  {
204
205      int num;
206      num = 12.12;
207
208
209
210
```

总结

```
184
185     a. 如果给的数据太大太大，直接报语法错误。
186     b. 如果我们为int变量赋值了1个小数，只会存储这个小数的整数部分。
187
188
```

验证

```
201 #include <stdio.h>
202
203 int main(int argc, const char * argv[])
204 {
205
206     int num;
207     num = 129992;
208
209
210
211     printf("%d\n", num);
212
213     printf("A\n");
214     return 0;
215 }
```

12
A
Program ended with exit code: 0

3>超过范围比较小

```
201 */
202 #include <stdio.h>
203
204 int main(int argc, const char * argv[])
205 {
206
207     int num;
208     num = 2200000000;
209
210
211
212     printf("%d\n", num);
213
214     printf("A\n");
215 }
```

-2094967296
A
Program ended with exit code: 0

183 因为变量的类型是int类型的, 所以为int变量赋值的时候, 赋值的数据不能超过指定的范围。

184

185 a. 如果给的数据太大太大, 直接报语法错误。

186 b. 如果我们为int变量赋值了1个小数, 只会存储这个小数的整数部分, 不会做四舍五入。

187 c. 如果给的数据超过的范围并不多, 这个时候存储的是随机值。

188

3.float型变量赋值注意点

1>

190

191 3). 为float类型的变量赋值。

192

193 a. 在C语言中如果我们直接写1个小数, 那么这个小数的类型是double类型的

194

195

196

现在给一个float变量用赋值double小数赋值, 会丢失精度

```

209 */
210 #include <stdio.h>
211
212 int main(int argc, const char * argv[])
213 {
214     float f1;
215
216     f1 = 12.12;
217

```

2>正确写法

```

209 */
210 #include <stdio.h>
211
212 int main(int argc, const char * argv[])
213 {
214
215     float f1;
216
217     f1 = 12.12f;

```

总结

```

191 3). 为float类型的变量赋值。
192
193 a. 在C语言中如果我们直接写1个小数。那么这个小数的类型是double类型的。
194    如果我们就是希望这个小数是1个float类型的。那么就在这个小数的后面加1个f/F
195    代表前面的这个小数的类型是1个float类型的。
196
197    记住这句话：为float变量赋值的时候。赋值的小数后面要加1个f
198
199

```

3>float的精度问题

```

191 3). 为float类型的变量赋值。
192
193 a. 在C语言中如果我们直接写1个小数。那么这个小数的类型是double类型的。
194    如果我们就是希望这个小数是1个float类型的。那么就在这个小数的后面加1个f/F
195    代表前面的这个小数的类型是1个float类型的。
196
197    记住这句话：为float变量赋值的时候。赋值的小数后面要加1个f
198
199
200 b. float只能精确保存有效位数不超过7位的小数。
201    如果赋值的数据的有效位数超过了7位。那么超出的部分就不会精确保存。
202
203

```

```

222 */
223 #include <stdio.h>
224
225 int main(int argc, const char * argv[])
226 {
227
228     float f1;
229
230     f1 = 12.123458723f;
231
232
233     printf("%f\n", f1);
234
235
236 //     int num;

```

```

12.123459
Program ended with exit code: 0

```

4.double型变量赋值注意点

204 4). 为double类型的变量赋值。

205
206 a. 在C语言中如果我们直接写1个小数,那么这个小数的类型是double类型的。
207 所以可以直接将1个小数赋值给1个double变量。

208
209 b. double只能精确保存有效位数不超过16位的小数。
210 如果赋值的数据的有效位数超过了16位,那么超出的部分就不会精确保存。
211

5.char型变量赋值注意点

1>

211
212 5). 为char类型的变量赋值。

213
214 a. char变量中只能存储单个字符,并且这单个字符还要用单引号引起来。

```
240 */  
241 #include <stdio.h>  
242  
243 int main(int argc, const char * argv[])  
244 {  
245  
246     char ch;  
247     ch = 'a';  
248  
249  
250
```

2>

213 5). 为char类型的变量赋值。

214
215 a. char变量中只能存储单个字符,并且这单个字符还要用单引号引起来。

216
217 b. 如果给的字符太多,保存最后1个。
218

```
243 #include <stdio.h>  
244  
245 int main(int argc, const char * argv[])  
246 {  
247  
248     char ch;  
249     ch = 'abcx';  
250  
251  
252     printf("%c\n",ch);  
253  
254  
255     // double d1;  
256     // d1 = 12.12;  
257  
258     // float f1;  
259     //  
260     // f1 = 12.123458723f;  
261     //  
262     //
```

Implicit conversion from 'int' to 'char' changes value from 1633837944 to 120

Program ended with exit code: 0

3>

c. 空格也是1个字符，所以也可以存储1个空格

```
245 #include <stdio.h>
246
247 int main(int argc, const char * argv[])
248 {
249     char ch;
250     ch = ' ';
251
252 }
```

4>

d. char变量中无法存储中文。

在C语言中1个中文字符占3个字节 而1个char变量占1个字节。

```
245 #include <stdio.h>
246
247 int main(int argc, const char * argv[])
248 {
249     char ch;
250     ch = '中';
251
252     printf("%c\n", ch);
253
254
255 }
```