C语言中堆内存分配通过malloc实现，通过free释放。

例：程序heap\_test

int\* func();

int main(int argc, char\* argv[])

{

int j;

int\* p = NULL;

p = func();

for (j = 0; j != 3; ++j)

{

printf("%d\n", p[j]);

}

free(p);

p = NULL;

return 0;

}

int\* func()

{

int i = 0;

int\* p = (int\*)malloc(3 \* sizeof(int));

for (i = 0; i != 3; ++i)

{

p[i] = i + 3;

}

return p; // 可以把堆内存当成返回值，但如果堆内存不是返回值，那么在函数结束后，分配的堆内存也无法在函数之外使用。

}

二重指针的堆内存分配：

例：程序heap\_test1

int\*\* func();

int main(int argc, char\* argv[])

{

int i;

int\*\* p = NULL;

p = func();

for (i = 0; i != 4; ++i)

{

printf("%d, ", p[0][i]);

}

printf("\n");

for (i = 0; i != 5; ++i)

{

printf("%d, ", p[1][i]);

}

printf("\n");

for (i = 0; i != 6; ++i)

{

printf("%d, ", p[2][i]);

}

printf("\n");

free(p[0]);

p[0] = NULL;

free(p[1]);

p[1] = NULL;

free(p[2]);

p[2] = NULL;

free(p);

p = NULL;

return 0;

}

int\*\* func()

{

int i;

// 指针数组，用二重指针表示

int\*\* p = (int\*\*)malloc(3 \* sizeof(int\*)); // p中可以存放3个指针变量

int\* p1 = (int\*)malloc(4 \* sizeof(int));

int\* p2 = (int\*)malloc(5 \* sizeof(int));

int\* p3 = (int\*)malloc(6 \* sizeof(int));

for (i = 0; i != 4; ++i)

{

p1[i] = i + 1;

}

for (i = 0; i != 5; ++i)

{

p2[i] = i + 2;

}

for (i = 0; i != 6; ++i)

{

p3[i] = i + 3;

}

p[0] = p1;

p[1] = p2;

p[2] = p3;

//free(p1); // 不能释放内存，否则，二重指针指向的是未知的地址

//free(p2);

//free(p3);

return p;

}

例：程序heap\_test2

#include <malloc.h>

#include <stdio.h>

int main(int argc, char\* argv[])

{

int i;

char\* p = (char\*)malloc(4 \* sizeof(char));

char\* p1 = (char\*)malloc(4 \* sizeof(char));

// 不能这样为堆内存的指针赋值

// ++p这个操作移动了p的初始地址

// 在后续执行free(p)时就会报错

//for (i = 0; i < 3; ++i)

//{

// \*p = 'a';

// ++p;

//}

//\*p = '\0';

// 可以像下面这样进行赋值

for (i = 0; i < 3; ++i)

{

\*(p + i) = 'a';

}

\*(p + 3) = '\0';

for (i = 0; i < 3; ++i)

{

p1[i] = 'b';

}

p1[i] = '\0';

printf("%s\n", p);

free(p);

p = NULL;

printf("%s\n", p1);

free(p1);

p1 = NULL;

return 0;

}