Malloc

Память для глобальных переменных выделяется во время компиляции. в стеке – для нестатических локальных переменных. Дополнительная память не выделяется во время выполнения программы.

Потребность выделения и освобождения памяти для структур данных.

Функция динамического распределенния данных находится в динамически распределяемой области памяти (heap) – свободная область памяти (резервная)

void \*malloc(size\_t size); \*\* возвращает адрес начала памяти куда можно написать. указатель типа void на size байт (первый байт неинициализированной памяти или NULL (если памяти недостаточно) . Резервирует память в байтах не менее size. void может быть чем угодно

void \*calloc(size\_t n, size\_t size) \*\*возвращает указатель на участок памяти, досточный для размещения n объектов

\*\*заданного размера size или Null (если недосточно памяти). Память инициализ нулями

int \*ip;

ip = (int \*), calloc(n, sizeof(int));

free(p) \*\*освобождение участка памяти на который указывает указатель р

for (p = head; p !=NULL; p = q)

{

q = p-> next;

free (p);

}

char \*p; \*\*указатель ссылается на первый из 1000 байтов выделенного участка памяти

p = malloc(1000);

int \*p;

p = malloc(50 \* sizeof (int) ); \*\*выделяется память для 50 целых

p = malloc(100); \*\* проверка выделения памяти

if (!p)

{

printf(“Нехватка памяти.\”);

exit(1); \*\*можно постаивть обработчик ошибок

}

void free(void \*p); \*\*возвращает системе выделенный ранее участок памяти

#include <stdio.h>

int \*get() \*\*функция которая находится в стеке. и возращает адресс

{

int i;

return (&i); \*\*функция отправит адрес на int

}

int \*set()

{

int lol;

lol = 78;

return (0); \*\*я беру адрес пространства памяти в стеке и расширяю его

}

int main(void)

{

int \*ptr;

ptr = get(); \*\*вы пытаетесь вернуть адрес который находитя в стеке

\*ptr = 12; \*\*адрес int стал равен 12

printf("%d\n", \*ptr); \*\*

set (); \*\* снова расширяет стек, инт получил значение 78м

printf("%d\n", \*ptr); \*\* в этом адресе стека теперь есть инт, и он получит значени 78

return (0);

}

Это есть ошибка! если мы вовращаем адрес стека, это адрес будет повторно использован в очень короткое время при вызове функции. В стека память возвращается автоматически.

malloc принимает параментр size\_t и размер size в байтах. отправит нам память ( int, float, char неважно)

void\* возвращает только 1 адрес начала памяти,

int \*tab;

tab = malloc(4); \*\*или выделили 1 int, если поменяю ОС – то код не будет работать

tab = malloc(sizeof(int)); \*\*или выделили память под 1 инт, но если поменяет int на long int, то работать не будет

tab = malloc(sizeof(\*tab)); \*\*или принимает переменную \*tab (tab указывает на тип переменной)

tab = malloc(sizeof(\*tab) \* 9); \*\*или выделяем память под 9 int

return (tab);

#include <stdio.h>

#include <stdlib.h>

int \*get()

{

int \*tab;

tab = malloc(sizeof(\*tab) \* 9);

return (tab);

int \*set()

{

int lol;

lol = 78;

return (0); \*\*я беру адрес пространства памяти в стеке и расширяю его

}

int main(void)

{

int \*ptr;

ptr = get(); \*\*вы пытаетесь вернуть адрес который находитя в стеке

\*ptr = 12; \*\*адрес int стал равен 12

printf("%d\n", \*ptr); \*\*

set (); \*\* снова расширяет стек, инт получил значение 78м

printf("%d\n", \*ptr); \*\* в этом адресе стека теперь есть инт, и он получит значени 78

printf("%p\n", \*ptr); \*адрес содержащий ptr

free(ptr); \*\*очистить память высвободить

return (0);

printf("%p\n", \*ptr); \*\*после free не может использовать, указатель на адрес не останется доступным

\*\*не должны заходить, чтобы увидеть снова

}

принимает адрес без парамента, который был возвращен , malloc раньше, и возвращает память

когда применяете free адрес больше не можете использовать, free принимает адресс в качестве парамента

free копирует только значение , которое является адресом