Задание №6

Выравнивание переменных. Представление в памяти структур

Расположение в памяти локальных переменных

# Описание нескольких локальных переменных разных типов.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <stdbool.h>  int main(void)  {  int a = 63;  long int l = 12345689;  double d = 123.312;  float f = 1231.750;  short s = 3451;  char c = 'T';  bool x = false;  size\_t z = 11111111111;    printf("%d **\t** %ld **\t** %f **\t** %lf **\t** %hi **\t** %c **\t** %zu **\t**", a, l, d, f, s, c, z);  printf(x ? "true" : "false");  printf("**\n**");  **return** EXIT\_SUCCESS;  } |

# Расположение в памяти

|  |
| --- |
| (gdb) x /36xb &c  0xffffffffeecc: 0x54 0x00 0x7b 0x0d 0x3f 0x00 0x00 0x00  0xffffffffeed4: 0x00 0xf8 0x99 0x44 0x59 0x61 0xbc 0x00  0xffffffffeedc: 0x00 0x00 0x00 0x00 0x87 0x16 0xd9 0xce  0xffffffffeee4: 0xf7 0xd3 0x5e 0x40 0xc7 0x19 0x46 0x96  0xffffffffeeec: 0x02 0x00 0x00 0x00  0xffffffffeed0: 0x3f 0x00 0x00 0x00 |

|  |  |  |
| --- | --- | --- |
| Имя переменной | Размер в байтах | Значение адреса |
| a | 4 | 0xffffffffeed0 |
| l | 8 | 0xffffffffeed8 |
| d | 8 | 0xffffffffeee0 |
| f | 4 | 0xffffffffeed4 |
| s | 2 | 0xffffffffeece |
| c | 1 | 0xffffffffeecc |
| x | 1 | 0xffffffffeecd |
| z | 8 | 0xffffffffeee8 |

# Зависимость значения адреса переменной от её размера.

Переменные записываются в памяти в порядке возрастания размера и объявлении в программе.

Первым делом записывается переменная “c”, которая имеет размер 1 байт и объявлена раньше переменной “x”. После этого идут переменные “x”, “s”, “a”, “f”, “l”, “d” и “z” по порядку. Переменные хранятся по порядку с начальным адресом “0xffffffffeecc” и конечным “0xffffffffeee8”. Каждая переменная имеет адрес вида “адрес предыдущей переменной + размер текущей переменной”.

Представление структур в памяти.

# Cтруктура с полями разного типа

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  **struct** date  {  short day;  int month;  long int year;  size\_t week;  float c\_temperature;  double f\_temperature;  };  int main(void)  {  **struct** date today =  {.day = 1, .month = 5, .year = 2022, .week = 17, .c\_temperature = 12.5, .f\_temperature = 54.68};  printf("Today's date is %hi.%d.%ld %zu-th week of this year.\  **\n**The weather is %.1lf degrees Celsius or %.3f degrees Fahrenheit",\  today.day, today.month, today.year, today.week, today.c\_temperature,\  today.f\_temperature);  printf("**\n**");    **return** EXIT\_SUCCESS;  } |

1. Поля структуры в памяти располагаются в порядке описания.

С целью оптимизации доступа компилятор может располагать поля в памяти не одно за другим, а по адресам кратным, например, размеру поля.

Адрес первого поля совпадает с адресом переменной структурного типа.

Поля структуры могут иметь любой тип, кроме типа этой же структуры, но могут быть указателями на него.

Дамп памяти , который содержит эту структуру.

|  |
| --- |
| (gdb) x /40xb &today  0xffffffffeec8: 0x01 0x00 0x00 0x00 0x05 0x00 0x00 0x00  0xffffffffeed0: 0xe6 0x07 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeed8: 0x11 0x00 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeee0: 0x00 0x00 0x48 0x41 0x00 0x00 0x00 0x00  0xffffffffeee8: 0xd7 0xa3 0x70 0x3d 0x0a 0x57 0x4b 0x40 |



|  |  |  |
| --- | --- | --- |
| Имя поля | Размер в байтах | Значение адреса |
| day | 2 | 0xffffffffeec8 |
| month | 4 | 0xffffffffeecc |
| year | 8 | 0xffffffffeed0 |
| week | 8 | 0xffffffffeed8 |
| c\_temperature | 4 | 0xffffffffeee0 |
| f\_temperature | 8 | 0xffffffffeee8 |

Поля структуры памяти располагаются в порядке описания. Зависимость адреса поля заключается в размере. Адрес первого поля совпадает с адресом переменной структурного типа. Адрес строки в котором описан указатель или в данном случае данные которые занимают 4 или 8 байт имеют свои особенности. Адрес указателя обязательно должен быть кратен 8 или 4 в зависимости от того 32-битная или 64-битная машина. В данном случае после строки day идут нулевые байты, до того момента пока адрес, как было сказано ранее не будет кратен размеру нашей строки. Так как все строки после первой имеют размер 4 или 8, то после второй строки их адреса будут идти друг за другом.

1. Адрес переменной структурного типа совпадает с адресом первого поля, в данном случае “0xffffffffeec8”.
2. Упакованная структура:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #pragma pack(push, 1)  **struct** date  {  short day;  int month;  long int year;  size\_t week;  float c\_temperature;  double f\_temperature;  };  #pragma pack(pop)  int main(void)  {  **struct** date today =  {.day = 1, .month = 5, .year = 2022, .week = 17, .c\_temperature = 12.5, .f\_temperature = 54.68};  printf("Today's date is %hi.%d.%ld %zu-th week of this year.\  **\n**The weather is %.1lf degrees Celsius or %.3f degrees Fahrenheit",\  today.day, today.month, today.year, today.week, today.c\_temperature,\  today.f\_temperature);  printf("**\n**");    **return** EXIT\_SUCCESS;  } |

Дамп памяти:

|  |
| --- |
| 0xffffffffeec8: 0x01 0x00 0x05 0x00 0x00 0x00 0xe6 0x07  0xffffffffeed0: 0x00 0x00 0x00 0x00 0x00 0x00 0x11 0x00  0xffffffffeed8: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeee0: 0x48 0x41 0xd7 0xa3 0x70 0x3d 0x0a 0x57  0xffffffffeee8: 0x4b 0x40 |

Размер структуры до упаковки составлял 40 байтов, после 34 байта.

|  |  |  |
| --- | --- | --- |
| Имя поля | Размер в байтах | Значение адреса |
| day | 2 | 0xffffffffeec8 |
| month | 4 | 0xffffffffeeca |
| year | 8 | 0xffffffffeece |
| week | 8 | 0xffffffffeed6 |
| c\_temperature | 4 | 0xffffffffeede |
| f\_temperature | 8 | 0xffffffffeee2 |

После упаковки строки стали распологаться с адресами от “0xffffffffeec8” до “0xffffffffeee2” друг за другом в порядке их описания, вне зависимости от размера, как было в структуре до упаковки.

Адрес переменной структурного типа совпадает с адресом первого поля, в данном случае “0xffffffffeec8”.

1. При перестановке полей структуры минимальное место занимаемой структурой равно 40 байтам. потому, что компилятор выравнивает переменные для быстрого доступа. Общий шаблон заключается в том, что когда базовый тип занимает N байт (где N - это степь 2, например, 1, 2, 4, 8, 16 - и редко больше), переменная должна быть выровнена по границе N-байта (кратная N байтам).
2. Добавим в программу массив элементов структурного типа:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  **struct** date  {  long int year;  size\_t week;srr  float c\_temperature;  short day;  double f\_temperature;  int month;  };  int main(void)  {  **struct** date arr[2] = {  { .day = 2, .month = 5, .year = 2022, .week = 17, .c\_temperature = 11.5, .f\_temperature = 60.50},  {.day = 3, .month = 5, .year = 2022, .week = 17, .c\_temperature = 15.5, .f\_temperature = 70.50},  };  **struct** date today =  {.day = 1, .month = 5, .year = 2022, .week = 17, .c\_temperature = 12.5, .f\_temperature = 54.68};  printf("Today's date is %hi.%d.%ld %zu-th week of this year.\  **\n**The weather is %.1lf degrees Celsius or %.3f degrees Fahrenheit",\  today.day, today.month, today.year, today.week, today.c\_temperature,\  today.f\_temperature);  printf("**\n**");  printf("%ld", **sizeof**(arr));    **return** EXIT\_SUCCESS;  } |

Выведем дамп памяти массива:

|  |
| --- |
| (gdb) p sizeof arr  $1 = 80  (gdb) x /80xb arr  0xffffffffee98: 0xe6 0x07 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeea0: 0x11 0x00 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeea8: 0x00 0x00 0x38 0x41 0x02 0x00 0x00 0x00  0xffffffffeeb0: 0x00 0x00 0x00 0x00 0x00 0x40 0x4e 0x40  0xffffffffeeb8: 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeec0: 0xe6 0x07 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeec8: 0x11 0x00 0x00 0x00 0x00 0x00 0x00 0x00  0xffffffffeed0: 0x00 0x00 0x78 0x41 0x03 0x00 0x00 0x00  0xffffffffeed8: 0x00 0x00 0x00 0x00 0x00 0xa0 0x51 0x40  0xffffffffeee0: 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 |

Завершающее выравнивание нужно для того , чтобы элементы массива (структуры) были правильно выравнены, чтобы процессору было проще найти.

В нашем случае есть завершающее выравнивание. Последний элемент структуры имеет тип “int” и после него идет 4 пустых байта.