

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

---

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА  
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Катедра «Комп'ютерна інженерія та програмування»

**ЗВІТ**

про виконання лабораторної роботи №4  
з навчальної дисципліни «Алгоритми та структури даних»

**Варіант 9**

Виконав студент:

Ульянов Кирило Юрійович

Група: КН-1023b

Перевірив:

старший викладач

Бульба С.С.

Харків-2024

# 1 Мета роботи

Отримати та закріпити знання про внутрішнє подання інтегрованих структур даних у мовах програмування.

## 2 Хід роботи

Написати програму, яка виводить на екран внутрішнє подання структури з варіантною частиною та з бітовими полями, а також масиву структур. Перелік властивостей та відповідні типи полів для об'єктів з табл. 4.1 обрати за своїм розсудом.

Дослідити, як виконується вирівнювання даних та полів структури. Порівняти час доступу до даних з вирівнюванням та за умови відсутності вирівнювання. За результатами роботи підготувати звіт з лабораторної роботи, де навести отримані результати та дати щодо них пояснення, зробити висновки.

9	Джерело струму	батареяка, акумулятор	– заряджено, – багаторазове використання
---	----------------	--------------------------	---

## 2.1 Ініціалізація структури

Розглянемо створену мною структуру даних для лабораторної, у ній використовуються як бітові поля, так і варіативна частина:

```

1  typedef enum { CHARGED, REPEATED_USE } Status;
2
3  typedef enum { BATTERY_TYPE, CHARGER_TYPE } PowerSourceType;
4
5  typedef struct {
6      unsigned int voltage : 10; // Напруга батареї в десятих частинах вольт (0 -
7                                // 1023, що дозволяє зберігати до 102.3 В)
8      unsigned int capacity : 12; // Ємність батареї в ампер-годинах (0 - 4095,
9                                // наприклад, для зберігання до 409.5 Ah)
10     unsigned int chargeLevel : 7; // Рівень заряду в відсотках (0-100)
11     unsigned int cellCount : 4; // Кількість осередків в батареї (до 15 осередків)
12     Status status;
13 } Battery;
14
15 typedef struct {
16     unsigned int voltage : 10; // Напруга зарядного пристрою в десятих частинах
17                               // вольт (0-1023, до 102.3 В)
18     unsigned int current : 10; // Струм зарядного пристрою в десятих частинах
19                               // ампера (0-1023, до 102.3 А)
20     unsigned int
21         chargeTime : 12; // Час зарядки в хвилинах (0-4095, до 4095 хвилин)
22     unsigned int maxCapacity : 12; // Максимальна ємність акумулятора, з яким
23                                   // можна працювати (0-4095 Ah)
24     unsigned int portCount : 4; // Кількість доступних зарядних портів (0-15)
25     Status status;
26 } Charger;
27
28 // Об'єднання батареї та зарядного пристрою у одну варіативну частину
29 typedef union {
30     Battery battery;
31     Charger charger;
32 } PowerUnion;
33
34 // Фінальна структура яку я буду досліджувати
35 struct power_source {
36     PowerSourceType
37         sourceType; // Тип джерела живлення (батарея або зарядний пристрій)
38     PowerUnion source; // Варіативна частина (батарея або зарядний пристрій)
39     float cost; // Вартість
40 };

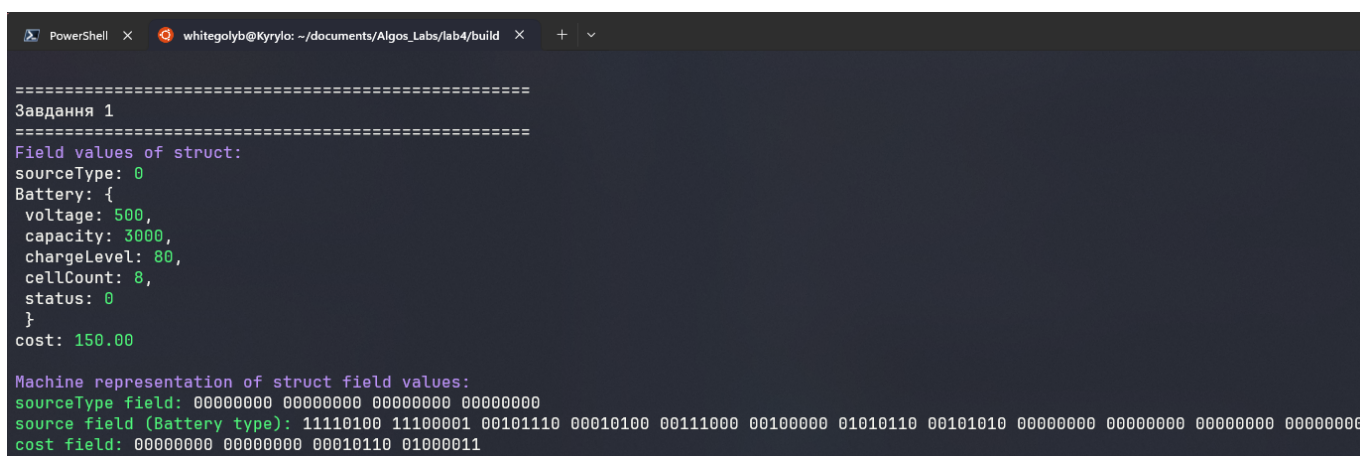
```

## 2.2 Машинний вигляд полів структури

Спочатку створю змінну типу моєї структури та заповню її, після чого виведу значення кожного поля та його машинне представлення:

```

1 void task1() {
2     struct power_source batterySource = {
3         BATTERY_TYPE, {.battery = {500, 3000, 80, 8, CHARGED}}, 150.0};
4
5     highlightText("Field values of struct: ", "blue");
6     print_struct_fields(&batterySource);
7     printf("\n");
8
9     highlightText("Machine representation of struct field values: ", "blue");
10    ;
11    print_binary_struct_fields(&batterySource);
12    printf("\n");
13 };
```



```

=====
Завдання 1
=====
Field values of struct:
sourceType: 0
Battery: {
  voltage: 500,
  capacity: 3000,
  chargeLevel: 80,
  cellCount: 8,
  status: 0
}
cost: 150.00

Machine representation of struct field values:
sourceType field: 00000000 00000000 00000000 00000000
source field (Battery type): 11110100 11100001 00101110 00010100 00111000 00100000 01010110 00101010 00000000 00000000 00000000 00000000
cost field: 00000000 00000000 00010110 01000011
```

Рис. 1. Машинне представлення кожного поля структури

Після чого виведу представлення всієї структури разом у машинному вигляді за допомогою функції **print\_binary\_struct**:

```

1 void print_binary_struct(void *ptr, size_t size) {
2     unsigned char *byte = (unsigned char *)ptr;
3     for (size_t i = 0; i < size; ++i) {
4         printBinary(byte[i]);
5     }
6     printf("\n");
7 }
8
9 void task1() {
10    struct power_source batterySource = {
11        BATTERY_TYPE, {.battery = {500, 3000, 80, 8, CHARGED}}, 150.0};
12
13    highlightText("Field values of struct: ", "blue");
14    print_struct_fields(&batterySource);
15    printf("\n");
16
17    highlightText("Machine representation of struct field values: ", "blue");
18    ;
19    print_binary_struct(ptr, sizeof(struct power_source));
20    printf("\n");
21 };
```

```

18     print_binary_struct_fields(&batterySource);
19     printf("\n");
20
21     highlightText("Machine representation of struct: ", "blue");
22     print_binary_struct(&batterySource, sizeof(batterySource));
23     printf("\n");
24 };

```

```

Machine representation of struct:
00000000 00000000 00000000 00000000 11110100 11100001 00101110 00010100 00111000 00100000 01010110 00101010
00000000 00000000 00000000 00000000 00000000 00000000 00010110 01000011

```

Рис. 2. Машинне представлення кожного поля структури

## 2.3 Машинне представлення цілої структури

Тепер створюю масив з 5 структур та виведу машинне представлення кожної структури в масиві структур у консоль:

```

1 void print_internal_int(int val) {
2     for (size_t i = 0; i < sizeof(int); i++) {
3         printBinary((unsigned char)(val >> (i * 8)));
4     }
5     printf("\n");
6 }
7
8 void print_internal_float(float val) {
9     unsigned char *floatBytes = (unsigned char *)&val;
10    for (int i = 0; i < sizeof(float); i++) {
11        printBinary(floatBytes[i]);
12    }
13    printf("\n");
14 }
15
16 void print_binary_struct(void *ptr, size_t size) {
17     unsigned char *byte = (unsigned char *)ptr;
18     for (size_t i = 0; i < size; ++i) {
19         printBinary(byte[i]);
20     }
21     printf("\n");
22 }
23
24 void print_binary_array_of_struct(const struct power_source *arr, int size) {
25     for (int i = 0; i < size; ++i) {
26         printf("\033[32m%d)\033[0m ", i + 1);
27         print_binary_struct(&arr[i], sizeof(arr[i]));
28     }
29     printf("\n");
30 }
31
32 void print_binary_struct_fields(const struct power_source *ps) {
33     printf("\033[32msourceType field: \033[0m");
34     print_internal_int(ps->sourceType);
35
36     if (ps->sourceType == BATTERY_TYPE) {

```

```

37     printf("\033[32msource field (Battery type): \033[0m");
38     print_binary_struct(&ps->source.battery, sizeof(ps->source.battery));
39 } else if (ps->sourceType == CHARGER_TYPE) {
40     printf("\033[32msource field (Charger type): \033[0m");
41     print_binary_struct(&ps->source.charger, sizeof(ps->source.charger));
42 }
43
44     printf("\033[32mcost field: \033[0m");
45     print_internal_float(ps->cost);
46 }
47
48 void print_struct_fields(const struct power_source *ps) {
49     printf("sourceType: \033[32m%d\033[0m\n", ps->sourceType);
50
51     if (ps->sourceType == BATTERY_TYPE) {
52         printf("Battery: { \n voltage: \033[32m%d\033[0m,\n capacity: "
53             "\033[32m%d\033[0m,\n "
54             "chargeLevel: \033[32m%d\033[0m,\n cellCount: "
55             "\033[32m%d\033[0m,\n status: \033[32m%d\033[0m\n }\n",
56             ps->source.battery.voltage, ps->source.battery.capacity,
57             ps->source.battery.chargeLevel, ps->source.battery.cellCount,
58             ps->source.battery.status);
59     } else if (ps->sourceType == CHARGER_TYPE) {
60         printf("Charger: { \n voltage: \033[32m%d\033[0m,\n current: "
61             "\033[32m%d\033[0m,\n chargeTime: \033[32m%d\033[0m,\n "
62             "maxCapacity: \033[32m%d\033[0m,\n portCount: "
63             "\033[32m%d\033[0m,\n status: \033[32m%d\033[0m\n }\n",
64             ps->source.charger.voltage, ps->source.charger.current,
65             ps->source.charger.chargeTime, ps->source.charger.maxCapacity,
66             ps->source.charger.portCount, ps->source.charger.status);
67     }
68
69     printf("cost: \033[32m%.2f\033[0m\n", ps->cost);
70 }
71
72 void task1() {
73     struct power_source batterySource = {
74         BATTERY_TYPE, {.battery = {500, 3000, 80, 8, CHARGED}}, 150.0};
75
76     struct power_source items[5] = {
77         {BATTERY_TYPE, {.battery = {600, 4000, 90, 10, CHARGED}}, 175.0},
78         {CHARGER_TYPE, {.charger = {250, 600, 100, 2000, 3, REPEATED_USE}},
79             60.0},
79         {BATTERY_TYPE, {.battery = {450, 2500, 60, 6, CHARGED}}, 120.0},
80         {CHARGER_TYPE, {.charger = {180, 400, 80, 1500, 2, REPEATED_USE}},
81             45.0},
81         {BATTERY_TYPE, {.battery = {550, 3500, 75, 7, CHARGED}}, 145.0}};
82
83     highlightText("Field values of struct: ", "blue");
84     print_struct_fields(&batterySource);
85     printf("\n");
86
87     highlightText("Machine representation of struct field values: ", "blue")
88     ;
89     print_binary_struct_fields(&batterySource);
90     printf("\n");
91
92     highlightText("Machine representation of struct: ", "blue");
93     print_binary_struct(&batterySource, sizeof(batterySource));
94     printf("\n");

```

```

94
95     highlightText("Machine representation of array with structs: ", "blue");
96     print_binary_array_of_struct(&items, 5);
97 };
```

```

Machine representation of array with structs:
1) 00000000 00000000 00000000 00000000 01011000 10000010 10111110 00010110 00111010 00000000 00000000 00000000 00000000 00000000 00000000 00101111 01000011
2) 00000001 00000000 00000000 00000000 11111010 01100000 01001001 00000110 11010000 00110111 00000000 00000000 00000001 00000000 00000000 00000000 01110000 01000010
3) 00000000 00000000 00000000 00000000 11000010 00010001 00100111 00001111 10101110 10110111 01111011 10011111 00000000 00000000 00000000 00000000 11110000 01000010
4) 00000001 00000000 00000000 00000000 10110100 01000000 00000110 00000101 11011100 00100101 00000000 00000000 00000001 00000000 00000000 00000000 00110100 01000010
5) 00000000 00000000 00000000 00000000 00100110 10110010 11110110 00010010 00000111 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00010001 01000011
```

Рис. 3. Машинне представлення масиву структур

```

=====
Завдання 1
=====
Field values of struct:
sourceType: 0
Battery: {
  voltage: 500,
  capacity: 3000,
  chargeLevel: 80,
  cellCount: 8,
  status: 0
}
cost: 150.00

Machine representation of struct field values:
sourceType field: 00000000 00000000 00000000 00000000
source field (Battery type): 11110100 11100001 00101110 00010100 00111000 00100000 01010110 00101010 00000000 00000000 00000000 00000000
cost field: 00000000 00000000 00010110 01000011

Machine representation of struct:
00000000 00000000 00000000 00000000 11110100 11100001 00101110 00010100 00111000 00100000 01010110 00101010 00000000 00000000 00000000 000
00000 00000000 00000000 00010110 01000011

Machine representation of array with structs:
1) 00000000 00000000 00000000 00000000 01011000 10000010 10111110 00010110 00111010 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00101111 01000011
2) 00000001 00000000 00000000 00000000 11111010 01100000 01001001 00000110 11010000 00110111 00000000 00000000 00000001 00000000 00000000 00000000
00000000 00000000 00000000 01110000 01000010
3) 00000000 00000000 00000000 00000000 11000010 00010001 00100111 00001111 10110110 10110111 01111011 10011111 00000000 00000000 00000000 00000000
00000000 00000000 00000000 11110000 01000010
4) 00000001 00000000 00000000 00000000 10110100 01000000 00000110 00000101 11011100 00100101 00000000 00000000 00000001 00000000 00000000 00000000
00000000 00000000 00000000 00110100 01000010
5) 00000000 00000000 00000000 00000000 00100110 10110010 11110110 00010010 00000111 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00010001 01000011
```

Рис. 4. Повний вивід результатів

## 2.4 Порівняння швидкості доступу до структур з вирівнюванням та без вирівнювання

Розмір структури з вирівнюванням становить 20 байт:

- **sourceType** - int 4 байта
- **source Batterytype** - структура з бітовими полями 12 байт
  - **voltage** - 10 біт + 22 біт доповнення = 4 байта
  - **capacity** - 12 біт + 4 біт доповнення = 2 байта
  - **chargeLevel** - 7 біт + 1 біт доповнення = 1 байта
  - **cellCount** - 4 біт + 4 біт доповнення = 1 байта
  - **status** - int 32 біта = 4 байта
- **cost** - float 4 байта

Для того щоб вимкнути вирівнювання на час досліду, до структур додаю додаткову строку у мові C:

```

1  typedef struct __attribute__((packed)) {
2      unsigned int voltage : 10; // Напруга батареї в десятих частинах вольт (0 -
3          // 1023, що дозволяє зберігати до 102.3 В)
4      unsigned int capacity : 12; // Їмність батареї в ампер-годинах (0 - 4095,
5          // наприклад, для зберігання до 409.5 Ah)
6      unsigned int chargeLevel : 7; // Рівень заряду в відсотках (0-100)
7      unsigned int cellCount : 4; // Кількість осередків в батареї (до 15 осередків)
8      Status status;
9  } Battery;
10
11 typedef struct __attribute__((packed)) {
12     unsigned int voltage : 10; // Напруга зарядного пристрою в десятих частинах
13         // вольт (0-1023, до 102.3 В)
14     unsigned int current : 10; // Струм зарядного пристрою в десятих частинах
15         // ампера (0-1023, до 102.3 А)
16     unsigned int
17         chargeTime : 12; // Час зарядки в хвилинах (0-4095, до 4095 хвилин)
18     unsigned int maxCapacity : 12; // Максимальна їмність акумулятора, з яким
19         // можна працювати (0-4095 Ah)
20     unsigned int portCount : 4; // Кількість доступних зарядних портів (0-15)
21     Status status;
22 } Charger;
23
24 // Фінальна структура яку я буду досліджувати
25 struct __attribute__((packed)) power_source {
26     PowerSourceType
27         sourceType; // Тип джерела живлення (батарея або зарядний пристрій)
28     PowerUnion source; // Варіативна частина (батарея або зарядний пристрій)
29     float cost; // Вартість
30 };

```

Рис. 5. Вимикання вирівнювання у структурах



Розмір структури **без вирівнювання** становить 18 байт:

- **sourceType** - int 4 байта
- **source** *Batterytype* - структура з бітовими полями 10 байт, тому що у PowerUnion саме більше поле це структура Charger - 10 байт без вирівнювання.
  - **voltage** - 10 біт
  - **capacity** - 12 біт
  - **chargeLevel** - 7 біт
  - **cellCount** - 4 біт
  - **status** - int 32 біта
- **cost** - float 4 байта

Тепер заміримо час доступу до полів структури **без вирівнювання та з ним**. Для отримання результатів я створив цикл що звертатиметься до її полів та запуску його мільярд разів, порахую середній час виконання:

```

1  #include "general_utils.h"
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  #define N 1000000000
7
8  void task1() {
9      struct power_source batterySource = {
10         BATTERY_TYPE, {.battery = {500, 3000, 80, 8, CHARGED}}, 150.0};
11
12     highlightText(
13         "Access time testing for structs with alignment and without it: ",
14         "blue");
15
16     clock_t start, end;
17     double cpu_time_used;
18
19     start = clock();
20
21     for (int i = 0; i < N; i++) {
22         unsigned int v = batterySource.source.battery.voltage;
23         unsigned int c = batterySource.source.battery.capacity;
24         unsigned int cl = batterySource.source.battery.chargeLevel;
25         unsigned int cc = batterySource.source.battery.cellCount;
26         int s = batterySource.source.battery.status;
27
28         (void)v;
29         (void)c;
30         (void)cl;
31         (void)cc;
32         (void)s;

```

```

33     }
34
35     end = clock();
36
37     cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
38
39     printf("Time for accessing struct fields (iterations %032m%d\033[0m):
40         "
41         "%.2f seconds\n",
42         N, cpu_time_used);

```

Результати роботи програми:

```

Access time testing for structs with alignment and without it:
Time for accessing struct fields (iterations 1000000000): 2.35 seconds
=====

```

Рис. 6. Структура без вирівнювання

```

Access time testing for structs with alignment and without it:
Time for accessing struct fields (iterations 1000000000): 2.21 seconds
=====

```

Рис. 7. Структура з вирівнюванням

### 3 Висновки

Під час виконання лабораторної роботи я навчився працювати зі структурами та представляти їх у машинному вигляді. Також розібрався з тим як працює вирівнювання на рівні компілятора та навіщо воно потрібно.

Після опрацювання отриманих результатів, можу зазначити що доступ до полів структури зі застосуванням вирівнювання здійснюється трішки швидше, ніж коли вирівнювання нема. Ці результати також можуть залежати від машини на якій відбувається код, але все ж таки бажано не вимикати вирівнювання задля забезпечення нормальної переносимості та сумісності програми з іншими пристроями, адже звертання до парних адрес комірок пам'яті відбувається швидше.