

Структури

Да разгледаме следната ситуация - искаме да създадем програма, в която да въвеждаме информацията за студентите в даден факултет. Всеки студент има следната информация - име, фамилия, факултетен номер и брой невзети изпити. Как би изглеждала една такава програма ?

```
#include<iostream>

int main()
{
    char firstname[50];
    char lastname[50];
    unsigned int faculty_number = 0;
    unsigned int failed_exams = 0;

    std::cout << "Enter first name: ";
    std::cin >> firstname;

    std::cout << "Enter last name: ";
    std::cin >> lastname;

    std::cout << "Enter faculty number: ";
    std::cin >> faculty_number;

    std::cout << "Enter failed exams: ";
    std::cin >> failed_exams;

    std::cout << firstname << " " << lastname << ", faculty number "
                << faculty_number << " has " << failed_exams << " failed exams" <<
    std::endl;
    return 0;
}
```

Как би изглеждала една такава програма, ако имаме повече от един студенти ?

Няма ли по-добър начин да напишем такава програма - да групираме всички променливи необходими за един студент в някакъв нов тип, който да е изграден от други типове ?

C++ ни позволява да групираме типове под ново име, като създаваме наш си, нов тип. Това се наричат структури (struct)

Как се дефинира за структура ?

```
struct име_на_структурата{
    тип1 име1;
    тип2 име2;
    тип3 име3;
    ...
};
```

Важно е да отбележим, че структурите се дефинират извън функциите, ако искаме да са видими в цялата програма.

Променливите вътре в структурите ще наричаме полета.

Можем да създаваме променливи от тип име_на_структурата по следния начин:

```
име_на_структурата име_на_променливата
```

Когато искаме да достъпим полета, използваме следния синтаксис:

```
име_на_променливата.име_на_полето
```

Примерът със студентите би изглеждал по този начин:

```
#include<iostream>

struct Student{
    char firstname[50];
    char lastname[50];
    unsigned int faculty_number = 0;
    unsigned int failed_exams = 0;
};

int main()
{
    Student one;
    std::cout << "Enter first name: ";
    std::cin >> one.firstname;

    std::cout << "Enter last name: ";
    std::cin >> one.lastname;

    std::cout << "Enter faculty number: ";
    std::cin >> one.faculty_number;

    std::cout << "Enter failed exams: ";
    std::cin >> one.failed_exams;

    std::cout << one.firstname << " " << one.lastname << ", faculty number "
                << one.faculty_number << " has " << one.failed_exams << " failed
exams" << std::endl;
    return 0;
}
```

Масиви от структури

Понеже структурите дефинират нов тип, съответно можем да направим масив от този тип. Ако искаме програмата за студентите да поддържа 50 студента, можем да си декларираме масив от тип `Student` по следния начин:

```
Student students[50]
```

Нека изнесем кода за въвеждане на информация за студент и този за извеждане на информацията в отделни функции.

```

#include<iostream>

struct Student{
    char firstname[50];
    char lastname[50];
    unsigned int faculty_number = 0;
    unsigned int failed_exams = 0;
};

Student create_from_user_input()
{
    Student result;
    std::cout << "Enter first name: ";
    std::cin >> result.firstname;

    std::cout << "Enter last name: ";
    std::cin >> result.lastname;

    std::cout << "Enter faculty number: ";
    std::cin >> result.faculty_number;

    std::cout << "Enter failed exams: ";
    std::cin >> result.failed_exams;
}

void print_student_info(Student student)
{
    std::cout << student.firstname << " " << student.lastname << ", faculty
number "
                << student.faculty_number << " has " << student.failed_exams << "
failed exams" << std::endl;
}

int main()
{
    Student students[50];

    for(int i = 0; i < 50; i++)
    {
        students[i] = create_from_user_input();
    }

    for(Student student : students)
    {
        print_student_info(student);
    }

    return 0;
}

```

Обръщам специално внимание на втория цикъл - той използва `for-each` синтаксис - подаваме име на променлива и масив, и цикъла обхожда масива, подобно на `for` цикъл.

Представяне в паметта

Колко място би заела следната структура ?

```
struct test{
    char a; //1B
    int b; //4B
    char c; //1B
};
```

Логичното нещо би било да е 6B

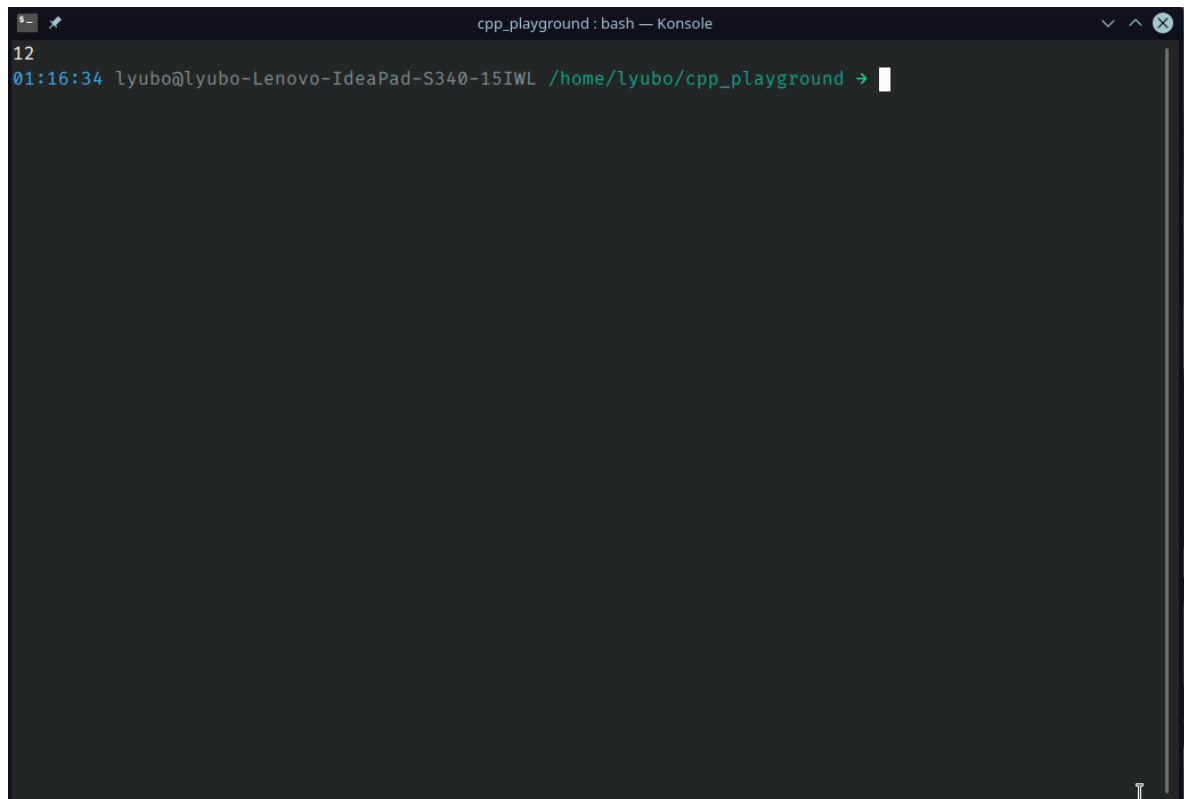
Нека проверим:

```
#include<iostream>

struct test{
    char a;
    int b;
    char c;
};

int main()
{
    test first;
    std::cout << sizeof(first) << std::endl;

    return 0;
}
```



```
cpp_playground : bash — Konsole
12
01:16:34 lyubo@lyubo-Lenovo-IdeaPad-S340-15IWL /home/lyubo/cpp_playground →
```

Когато пишем структури в C++, компилатора "подрежда" нашите структури, като всяко поле има адрес кратен на най-големия елемент.

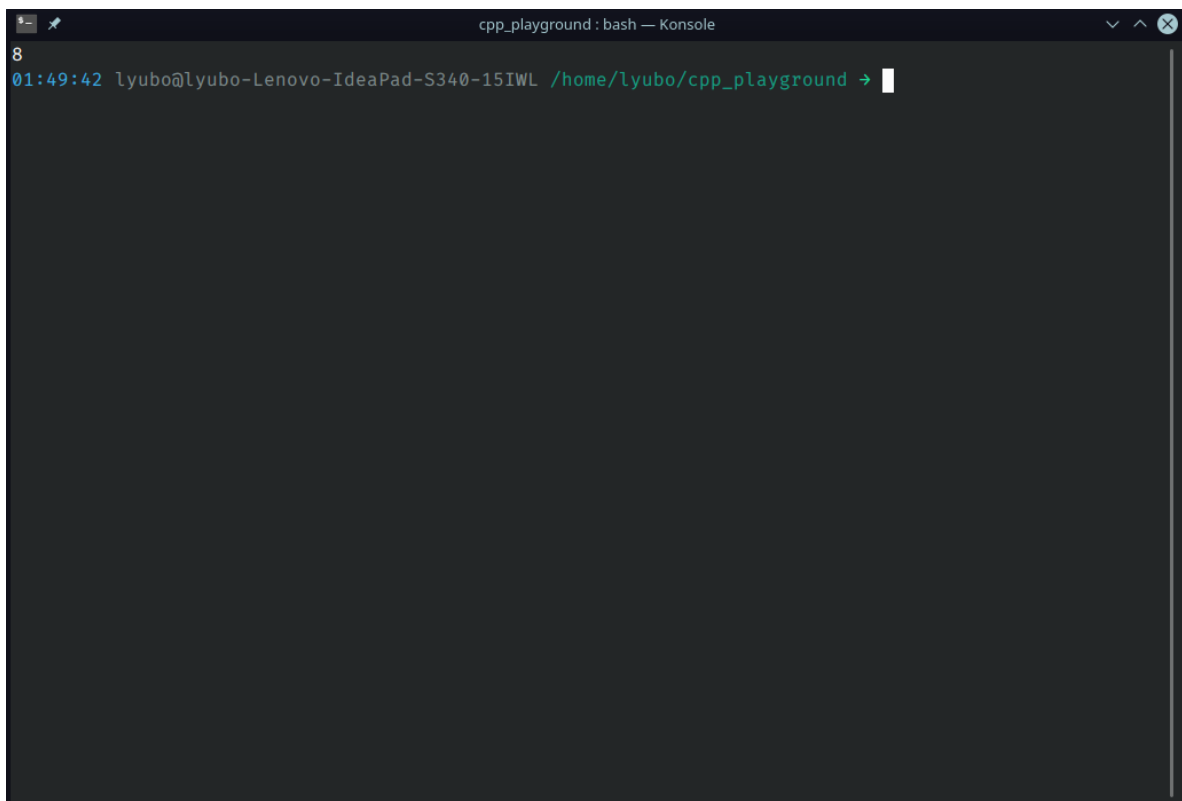


Понеже най-големия елемент в нашия случай е `int`, с размер 4B, елементите трябва да имат адреси кратни на 4B. Затова `char a` и `char c` заемат по 4B.

Нека разместим местата на `int b` и `char c`

```
#include<iostream>
struct test2{
    char a;
    char c;
    int b;
};

int main()
{
    test2 second;
    std::cout << sizeof(second) << std::endl;
    return 0;
}
```



```
cpp_playground : bash — Konsole
8
01:49:42 lyubo@lyubo-Lenovo-IdeaPad-S340-15IWL /home/lyubo/cpp_playground →
```



Тук отново най-големият тип е 4B, но понеже двете `char` променливи са една до друга, се подравняват само веднъж.

Ще разгледаме още един пример

```
#include<iostream>

struct test{
    char a;
    int b;
    char c;
};

struct test2{
    char a;
    char c;
    int b;
};

struct test3
{
    int b;
    char a;
    char c;
};

int main()
{
    test first;
    test2 second;
    test3 third;

    std::cout << sizeof(first) << std::endl;
    std::cout << sizeof(second) << std::endl;
    std::cout << sizeof(third) << std::endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
12
8
8
02:02:35 lyubo@lyubo-Lenovo-IdeaPad-S340-15IWL /home/lyubo/cpp_playground →
```

Правило: Когато подреждаме променливите в структура, винаги трябва да ги подреждаме спрямо техния размер - или като тръгнем от най-малките към най-големите, или от най-големите към най-малките.

Задачи

1. Напишете програма, която да обработва триъгълници. Напишете подходяща структура. Напишете следните функции - проверка дали триъгълника е валиден, функция за изчисляване на лице на триъгълник и функция за проверка, която приема два триъгълника и показва дали първия триъгълник има по-голям периметър.
 2. Напишете програма съдържа информация за студенти - име (до 50 символа), фамилия (до 50 символа), факултетен номер, оценки по предмети (5 предмета). Напишете функции за въвеждане, извеждане и пресмятане на среден успех на студент. Създайте 5 студента, намерете средният им успех по всички предмети и изведете студента с най-висок среден успех.
 3. Напишете програма, която съдържа информация за автомобили - марка (до 50 символа), модел (до 50 символа), конски сили, среден разход и цена. Направете функция за въвеждане. Нека потребителя число n, след това да въведе n коли. На екрана да се изведат колата с най-нисък среден разход и колата с най-добро съотношение цена/разход.
-