Ссылки. Пространства имён

Урок №34

ЧТО НАС ЖДЕТ?

ПЛАН УРОКА

- Теория: ссылки
- Решаем задачки
- Разбор домашней работы
- Разрабатываем новый EduApp
- Теория: пространства имён
- Решаем задачки
- Подводим итоги

Вспоминаем указатели

Указатели

Объявление:

```
int *имя = &переменная;
int a = 5;
int *ptr_a = &a;
```

Обращение:

```
*ptr_a = 6;
```

\- неудобно

Указатели - наследие языка Си.

Использование указателей не всегда удобно и порождает большое количество синтаксических ошибок

Теория

10 минут

Ссылки

Ссылки

Ссылка (англ. reference) - это псевдоним переменной

Объявление:

```
тип &имя = переменная;
```

Пример:

```
int a = 5;
int &b = a;     -   переменная a теперь имеет псевдоним b
b = 6;
cout << a;      -   на экран выведется 6
```

Отличия от указателей

Отличия от указателей

Благодаря ссылкам, вы обеспечиваете:

- безопасность
- простоту (не нужен оператор *)
- хорошую читаемость кода

С помощью ссылок вы **не можете:**

- получить адрес ссылки
- сравнить значения ссылок
- выполнить арифметические операции над ссылкой
- изменить ссылку

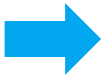
Ссылки и функции

Ссылки и функции

Передача через указатели

```
void swap_values(float *a, float *b){
    ...
}

swap_values(&var_1, &var_2);
```



Передача через ссылки

```
void swap_values(float& a, float& b){
    ...
}

swap_values(var_1, var_2);
```

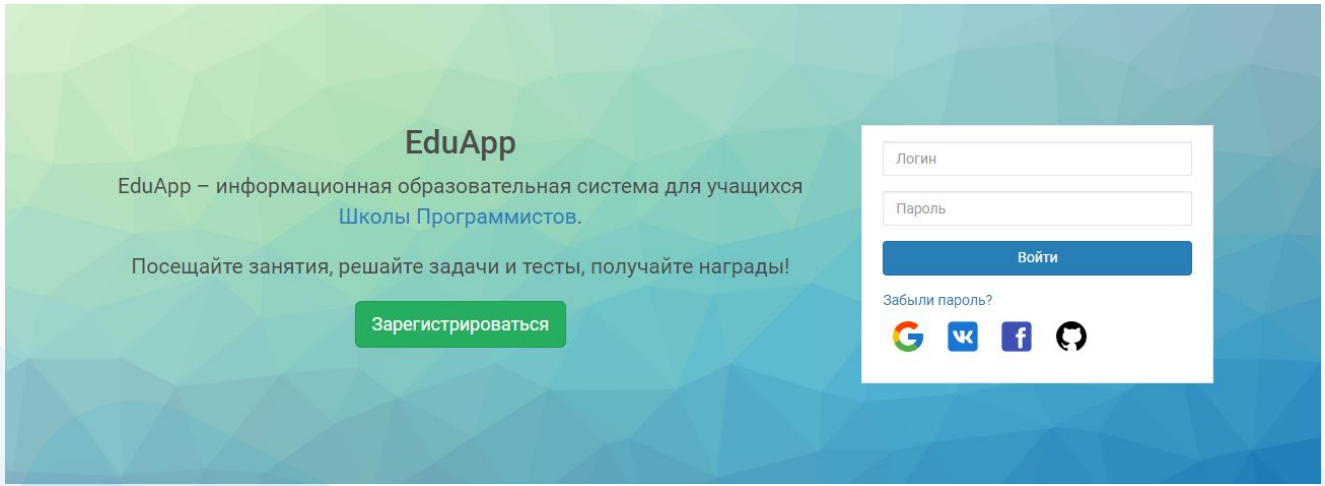
Практика

10 минут

Возвращаемся к структурам!

Разбор домашней работы

Приступаем к разработке EduApp



Практика

30 минут

Теория

10 минут

Проблема

При написании структур **Student** и **StudentJournal**, мы добавляли в код вспомогательные функции, которые работали с этими структурами...

```
Student read_student();
double get_student_avg_mark(Student person);
bool will_graduate(Student person, int after_years);
Student read_student();
bool equal_students(Student s1, Student s2);
void add_total_marks(StudentJournal &journal);
void filter_agv_above(StudentJournal &journal, double x = 4.5);
void remove_duplicates(StudentJournal &journal);
```

Проблема

При написании структур **Student** и **StudentJournal**, мы добавляли в код вспомогательные функции, которые работали с этими структурами...

```
Student read_student();
double get_student_avg_mark(Student person);
bool will_graduate(Student person, int after_years);
Student read_student();
bool equal_students(Student s1, Student s2);
void add_total_marks(StudentJournal &journal);
void filter_agv_above(StudentJournal &journal, double x = 4.5);
void remove_duplicates(StudentJournal &journal);
```

Удобно бы было читать код, если бы функций было 1000?

Проблема

При написании структур **Student** и **StudentJournal**, мы добавляли в код вспомогательные функции, которые работали с этими структурами...

Student

```
Student read_student();
double get_student_avg_mark(Student person);
bool will_graduate(Student person, int after_years);
Student read_student();
bool equal_students(Student s1, Student s2);
```

StudentJournal

```
void add_total_marks(StudentJournal &journal);
void filter_agv_above(StudentJournal &journal, double x = 4.5);
void remove_duplicates(StudentJournal &journal);
```

Удобно бы было читать код, если бы функций было 1000?

Как можно их разделять по принадлежности к той или иной сущности (Ученик, Журнал, Родитель, Курс)?

Пространства имен

Пространство имен

Namespace - абстрактное множество, созданное для логической группировки уникальных идентификаторов

Синтаксис:

```
namespace имя{
    элементы;
}
```

Пример:

```
namespace name {
    int variable;
    void func(string s){cout << s;}
}
name::variable = 5;
name::func("Hello");
```

Особенности namespace

- `using namespace имя` – упрощение доступа к элементам конкретного пространства пространства
  (после данной строки ко всем элементам пространства `имя` можно будет обращаться без `имя::`)

- `using имя::элемент` – упрощение доступа к конкретному элементу
  (после данной строки к `элемент` можно будет обращаться без `имя::элемент`)

- пространства имен могут быть вложенными

Практика

10 минут

Итоги урока

На занятии я **узнал** ....

На занятии я **понял** ....

На занятии я **сделал** ....

Продолжи любую фразу