OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Типы данных

Плисенко Ольга

Преподаватель курса «Программист С»

Эл. почта plolga.otus@yandex.ru
Telegram: @PlisenkoOlga

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в Telegram @OTUS C-2023-07

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

- Классические типы данных
- Структуры
- Объединения
- Перечисления
- Рефлексия

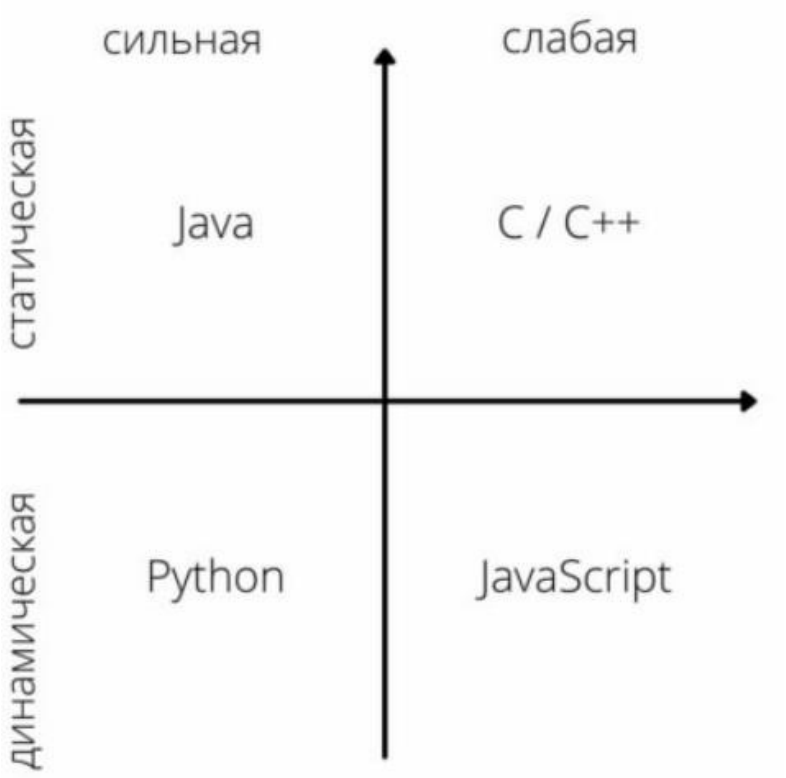
Цели вебинара

После занятия вы сможете

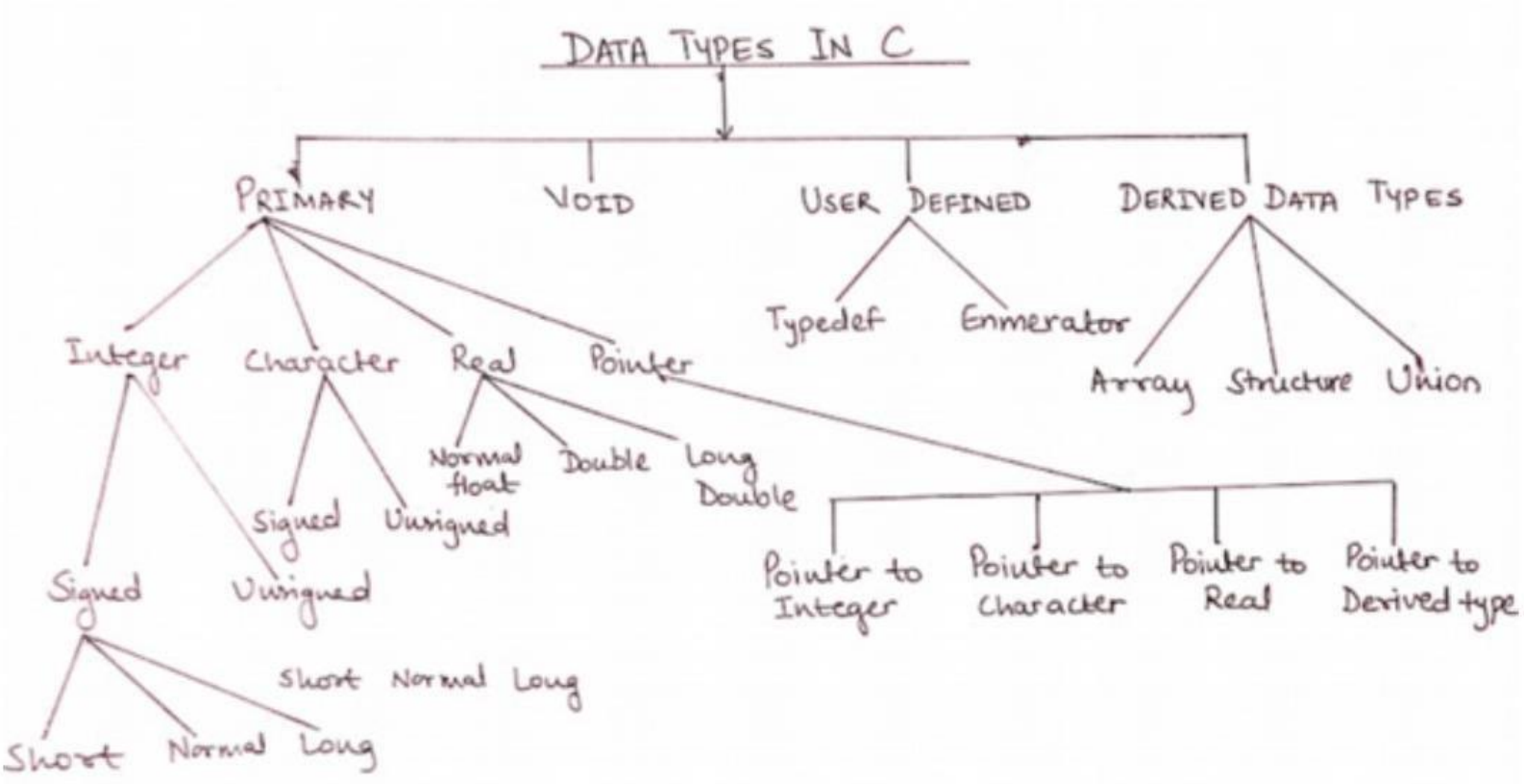
1. Вспомнить классические типы данных
2. Создавать собственные типы данных
3. Понять, как различные типы хранятся в памяти

Типы данных

C - язык с явной, слабой статической типизацией



Типы данных в C



Ключевые слова для типов данных

Стандарт K&R C	Добавления в стандарте C90	Добавления в стандарте C99
int	signed	_Bool
char	void	_Complex
float		_Imaginary
double		
short		
long		
unsigned		

Целочисленные константы

10 - число типа int

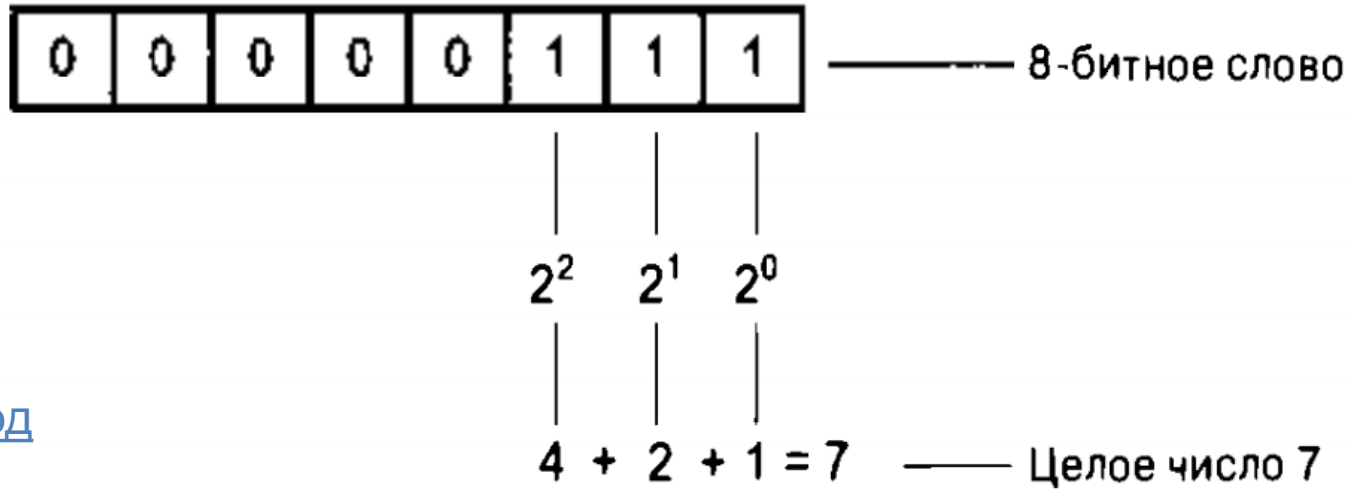
10L - число типа long

10LL - число типа long long

10ULL - число типа unsigned long long

Формат целого типа

Int:	%d	%u	%o	%x
long:	%ld	%lu	%lo	%lx
long long:	%lld	%llu	%llo	%llx
short:	%hd	%hu	%ho	%hx



[Дополнительный код]

Inttypes.h

int8_t - целочисленный тип с точной шириной

int_least8_t - тип с минимальной шириной

int_fast8_t - высокоскоростной тип с минимальной шириной

intmax_t - максимально возможный целочисленный тип

PRId32 - оптимальный спецификатор

float / double

float (32 бит: 8 бит экспонента и знак, 24 бит - мантисса) – 6 первых значащих цифр

double (64 бит) – 13 первых значащих цифр

long double – точность не уступает точности double

IEEE 754

Константы с плавающей запятой

мантисса e(E) знак порядок -5.4e5

C99: 0xb.2ep10 = (11 + 2/16 + 14/256) * 1024

где b = 11

2e = 2/16 + 14/256

p10 = 210 = 1024

5.0 - double

5.0f - float

5.0L - long double

float fl = 4.0 * 5.0;

Формат вещественных чисел

float / double	%f – десятичное представление %e – экспоненциальное представление %a – шестнадцатеричное представление		
long double	%Lf	%Le	%La

Floating point numbers are a leaky abstraction

Oracle - About Floating-Point Arithmetic

Avoiding Overflow, Underflow, and Loss of Precision

Библиотека GMP

GMP (GNU Multiple Precision Arithmetic Library) – бесплатная библиотека для «длинной арифметики»:

- поддержка целых чисел неограниченного размера (около 140 арифметических и логических функций);
- поддержка рациональных чисел неограниченного размера (около 35 функций, плюс можно применять функции для целых по отдельности к числителю и к знаменателю);
- поддержка действительных чисел произвольной точности (около 65 функций);
- быстро работает как с большими, так и с не очень большими операндами.

Применение: криптография, безопасность в Интернете (библиотека OpenSSL), системы алгебраических вычислений

Библиотека GMP

```
#include "gmp.h"
void fact(int n)
{
 int i;
 mpz_t p;
 mpz_init_set_ui(p, 1);             /* p = 1 */
 for(i = 1; i <= n; i++)
 {
                mpz_mul_ui(p, p, i); /* p = p * i */
 }
 printf ("%d! = ", n);
 mpz_out_str(stdout, 10, p);
 mpz_clear(p);
}
```

Пользовательские типы данных

1. Структура
2. Объединение
3. Перечисление
4. typedef

Структура

Структура – это набор переменных (обычно разного типа), объединенных одним именем.

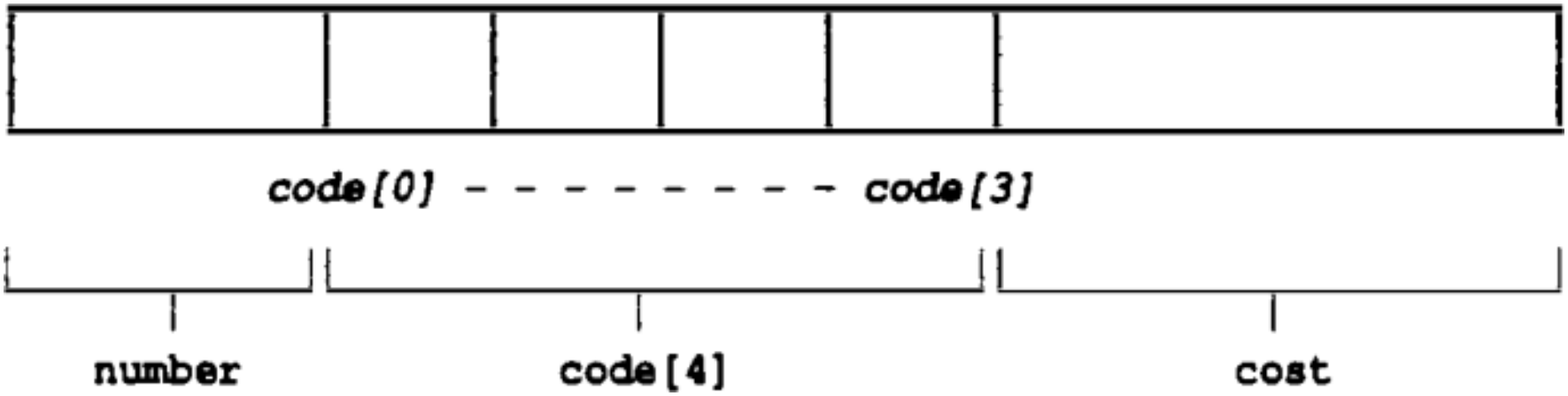
```
struct имя_типа_структуры {
тип имя_переменной_1;
…
тип имя_переменной_N;
} имя_экземпляра_структуры;
```

Доступ к переменным:

```
имя_экземпляра_структуры.имя_переменной
```

Структура

```
struct student {
        int number;
        char code[4];
        float cost;
}
struct student otus;
```



Структура

Инициализация:

```
struct student otus = {2021, “Gr_C”, 4.5};
```

Назначенная инициализация:

```
struct student otus = {.cost = 3.0, .code = “gr_C”, 4.5};
```

Доступ:

```
otus.number
```

Массив структур

Массив структур:
struct student otus[30];
otus[i].number

Динамический массив структур:
struct student *otus;
otus = (struct student*) malloc (count_rec * sizeof(struct student)); otus->number
(*otus).number

Передача структур в функцию

1. Передача частей структуры в функцию:
   f(otus.number);
   f(otus.code);
   f(otus.code[3]);

2.Передача структуры в качестве аргумента:
   int f_num(struct student o)
   f_num(otus);
   !!! Структура копируется в стек функции !!!

3. Передача структуры через указатель
   int f_num(struct student *o)
   f_num(&otus);

Структура или указатель на структуру???

struct vector {double x, double y};
1)
struct vector rez, a, b;
struct vector sum_vect(struct vector, struct vector);
…
rez = sum_vect(a, b);

2)
struct vector rez, a, b;
void sum_vect(struct vector *, struct vector *, struct vector *);
…
sum_vect(&a, &b, &rez);

Еще о структурах

1. Вложенность структур:
struct names {char first_n[25], char last_n[50] };
struct student{struct names fio, int year_b, …};

2. Инициализация и присвоение одной структуры другой:
struct names fio1 = {“Иван”, “Иванов”}
struct names fio2 = fio1;
или
fio2 = fio1;

И еще о структурах

struct names {char first_n[25], char last_n[50] };

или

struct names {char *first_n, char *last_n};

???

Объединение

Объединение – это тип данных, который позволяет хранить переменные разных типов в одном и том же месте памяти (но не одновременно).

```
union имя_типа {
        тип     имя_элемента_1;
…
        тип     имя_элемента_N;
}
```

Инициализация объединений

```
union types{
        int digit;
        double bigNum; char
        latter;};

union types p_char;

 p_char.latter = ‘A’;
union types p_char2 = p_char; //инициализация другим объединением
union types p_int = {50}; // инициализация первого элемента
union types p_dbl = {.bigNum = 50.0}; // назначенный инициализатор
```

Перечисление

Перечисляемый тип – используется для объявления символьных имен, соответствующих целочисленным константам.

enum имя_типа {имя1 = значение1, имя2 = значение2, …} переменная

enum spectr{red, orange, yellow, green, blue};

enum spectr color;

enum levels {low = 1, medium = 5, high = 10};

enum enimals{cat, dog = 10, puma, tiger};

typedef

Оператор typedef позволяет связать новые типы данных с существующими.

typedef существующий_тип новый_тип;

- выполняется компилятором;
- область видимости зависит от местоположения;
- используется при создании удобных имен для часто используемых типов.

1) typedef char * STRING;

STRING p1, p2; char * p1, * p2;

или

#define STRING char *

STRING p1, p2; char * p1, p2;

2) typedef struct{float real; float imag;} COMPLEX;

Рефлексия

Рефлексия

1. Назовите классические типы данных?
2. Перечислите пользовательские типы данных?
3. В чем разница между структурой и объединением?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Плисенко Ольга

Эл. почта plolga.otus@yandex.ru
Telegram: @PlisenkoOlga