Základy programovania 25ZS C++

Prednášajúci: prof. Gabriel Juhás
Cvičiaci: Milan Mladoniczky

C vs C++

- Procedurálny jazyk.
- Štandardné knižnice sú sústredia na funkcie nízkej úrovne.
- Manuálna správa pamäte (malloc, free).
- Ošetrenie chybových stavov plne v rukách vývojára.
- Definovanie makier cez #define.

- Multi-paradigmový jazyk, najmä objektovo-orientovaný prístup a generické programovanie.
- Bohatšia množina štandardných knižníc.
- Práca s pamäťou cez new a delete, podpora automatizovanej správy pamäte (garbage collection).
- Zahrnuté ošetrenie chýb (exception).

C vs C++

- Nie je možné preťažiť či upraviť existujúci kód.
- Konflikty a logické delenie kódu ponechané na programátora.
- Vstupno-výstupné funkcie scanf, printf ...

- Preťaženie operátorov a funkcii.
- Logické delenie programu do tzv. namespace.
- Podpora vstupno-výstupných streamov (cin, cout).
- Podpora generického programovania.
- Spätná kompatibilita s C.

C vs C++

Vhodné využitie:

- potreba prístupu k nižším úrovniam počítača.
- programovanie embedovných systémov

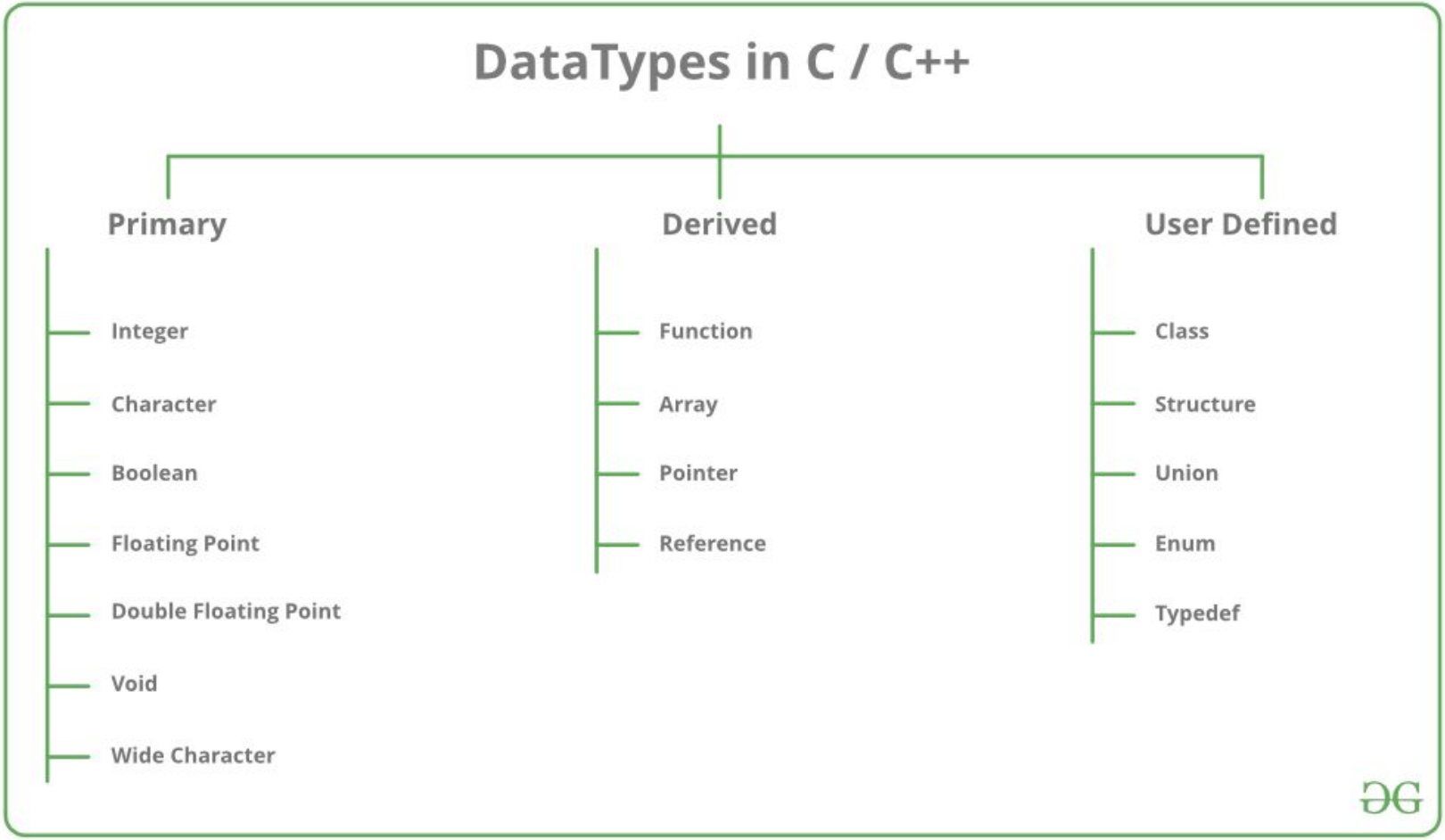
Vhodné využitie:

- komplexnejšie systém
- herný priemysel
- spracovanie dát v reálnom čase
- a mnoho ďalšieho

C++ základy

- Program rozdelený do namespace
- Podpora streamov
- Podpora typu string
- Kompatibilita s jazykom C

```
#include <iostream>

using namespace std;

int main() {

     int number = 24;

     string text = “Hello World”;

     cout << text << number << endl;

     return 0;

}
```


DataTypes in C / C++
Primary
Integer
Character
Boolean
Floating Point
Double Floating Point
Void
Wide Character
Derived
Function
Array
Pointer
Reference
User Defined
Class
Structure
Union
Enum
Typedef

Trieda - Class

- Programátorom definovaný typ
- Základný blok objektovo-orientovaných programov
- Bohatšie správanie ako štruktúra v C
- Z triedy sa vytvárajú objekty - alokovaná pamäť s atribútmi triedy

```
#include <iostream>

using namespace std;

class Person {

private:

        string name;

        int id;

public:

        void getDetails() {

                cout << "[" << id <<"]" name << endl;

        }

};

int main() {

        Person* p1 = new Person();

        p1->getDetails();

        delete p1;

        return 0;

}
```

Konštruktor

- Východzí (Default) prázdny konštruktor
- Parametrizovaný
- Konštruktor s inicializáciou členov triedy
- Copy konštruktor

```
#include <iostream>

using namespace std;

class MojaTrieda {

public:

        int id;

        MojaTrieda(int a): id(a) {}

};

int main() {

        MojaTrieda* objekt = new MojaTrieda(25);

        cout << objekt->id << endl;

        delete objekt;

        return 0;

}
```

Polymorfizmus

- Definovanie funkcie s rovnakým názvom ale rôznymi parametrami.
- Preťaženie operátorov pre vlastné správanie v rámci triedy.
- Preťaženie funkcií pri dedení
    - ak rodičovská trieda umožní preťaženie triedy pomocou kľúčového slova *virtual*

Dedenie

- Derivovanie triedy od druhej triedy.
- Vytvorenie vzťahu rodič-dieťa medzi triedami.
- Prenesenie funkcií a členov triedy.
- Lepšia enkapsulácia a abstrakcia logiky programu.
- Možné dodefinovať či sa dedí iba public časť, alebo private časť.

Praktická ukážka –>

https://zapr.interes.group/exercises/exercise-9

Doplňujúci materiál

- https://citeschool.org/sk/c-vs-c-39-hlavnych-rozdielov-medzi-jazykmi-c-a-c-s-prikladmi
- https://www.geeksforgeeks.org/c-classes-and-objects/
- https://www.geeksforgeeks.org/cpp-polymorphism/?ref=shm
- https://www.geeksforgeeks.org/inheritance-in-c/?ref=shm