# Обектно-ориентирано програмиране 2024/2025 г., спец. Софтуерно инженерство, група 5 Контролна работа No. 3

Задача 1:

Реализирайте на език C++ следната йерархия от класове:

* + Абстрактен базов клас EventBase, съдържащ следните чисто виртуални функции: print(), serialize()
  + Шаблонен клас Event<T>, който наследява EventBase и съдържа поле T payload. Реализира методите:
    - print() – извежда: Event: <payload>
    - serialize() – връща сериализиран payload(payload като my\_string)
  + Направете специализация на Event<my\_string>, така че:
    - serialize() да връща кавички около текста: "payload"
  + Реализирайте три наследници на Event<T>:
    - LogEvent<T>: добавя префикс "Log: " при print() и сериализация log=<...>
    - MetricEvent<T>: добавя префикс "Metric: ", сериализация metric=<...>
    - AlarmEvent<T>: добавя префикс "ALARM: ", сериализация ALARM=<...>

Използвайки тази йерархия, реализирайте хетерохенен контейнер по следния начин:

* Шаблонен клас EventContainer, който съхранява до capacity на брой указатели към EventBase (определянето на capacity става по време на компилация)
* Да има следнатите методи:
  + add– добавя копие от подаденото събитие.
  + pop– премахва последното събитие.
  + print\_all– извежда всички събития с техните print() и serialize().

Примерен main.cpp:

```

int main() {

EventContainer<10> container;

container.add(LogEvent<my\_string>("System started"));

container.add(MetricEvent<int>(55));

container.add(MetricEvent<my\_string>("Disk I/O"));

container.add(AlarmEvent<double>(91.2));

const AlarmEvent<my\_string> a("Disk full.");

container.add(a);

container.print\_all();

return 0;

}

```

Изход на примерната програма:

```

Log: Event: "System started"

Serialized : log = <"System started">

Metric : Event : 55

Serialized : metric = <55>

Metric : Event : "Disk I/O"

Serialized : metric = <"Disk I/O">

ALARM : Event : 91.2

Serialized : ALARM = <91.2>

ALARM : Event : "Disk full."

Serialized : ALARM = <"Disk full.">

```

Допълнителни указания:

* Пишете допълнителни функции, ако има нужда. Описаните в условието не са гаранция за пълно решение.
* include-ване и ползване на my\_string е позволено (имплементиран по време на упражнения).
* За превръщане на произволен тип в const char\*, може да използвате следния код:

```

#include <sstream>

double test = 10.1;

ostringstream oss;

oss << test;

const char\* str = oss.str().c\_str();

```