**2023 11 24**

**array decay**

***// array decay***

**int** main**()**

**{**

**using** **namespace** std**;**

array a **{**1**,** 3**,** 5**,** 1**,** 23**};**

**int** **\***ptr **=** a**;** ***// sytanx hatası (array decay olmuyor)***

**int** **\***p1 **=** a**.**data**();**

**int** **\***p2 **=** **&**a**[**0**];**

**int** **\***p3 **=** **&\***a**.**begin**();**

**int** a**[**5**]{};**

a**[**2**];** ***// \*(a + 2)***

3**[**a**]++;** ***// \*(a + 3) ++;***

array**<int,** 5**>** b**{};**

***// 3[a]++; gecersiz***

**}**

**std::tuple**

Farklı veri tiplerini bir arada tutmak için kullanılır

#include<tuple>

**int** main**()**

**{**

***// bir container değil***

std**::**tuple**<>** t0**;**

std**::**tuple**<int>** t1**;**

std**::**tuple**<int,** **double>** t2**;**

std**::**tuple**<int,** **double,** **long>** t3**;**

**}**

**int** main**()**

**{**

***// default ctor ya da zero init eder***

std**::**tuple**<int,** **double,** **long,** Date**>** tp**;**

cout **<<** get**<**0**>(**tp**)** **<<** "\n"**;** ***// int& döndürür 0***

cout **<<** get**<**3**>(**tp**)** **<<** "\n"**;** ***// 1 Ocak 1900***

cout **<<** **++**get**<**3**>(**tp**)** **<<** "\n"**;** ***// 2 Ocak 1900***

std**::**tuple**<int,** **double,** **long>** tp1 **=** **{**4**,** 4.5**,** 45L**};**

std**::**tuple tp2**{**4**,** 4.5**,** 45L**};** ***// CTAD***

**}**

**std::make\_tuple**

***// make\_tuple***

**template** **<typename** **...**Args**>**

**auto** MakeTuple**(**Args **&&** **...**args**)**

**{**

**return** std**::**tuple**<**Args**>(**std**::**forward**<**Args**>(**args**)...);**

**}**

**int** main**()**

**{**

**int** x **=** 235**;**

**double** dval **=** 3.4**;**

**char** c **=** 'A'**;**

autp tp **=** make\_tuple**(**x**,** dval**,** c**);**

cout **<<** get**<**0**>(**tp**);** **<<** "\n"**;**

cout **<<** get**<**1**>(**tp**);** **<<** "\n"**;**

cout **<<** get**<**2**>(**tp**);** **<<** "\n"**;**

cout **<<** get**<int>(**tp**);** **<<** "\n"**;** ***// eğer birden fazla int varsa ambiguity olr***

**}**

***// tuple --enum***

**int** main**()**

**{**

**using** **namespace** std**;**

**enum** **{**AGE**,** NAME**,** ID**};**

**using** age **=** **int;**

**using** name **=** std**::**string**;**

**using** id **=** **long;**

tuple**<**age**,** name**,** id**>** tp**{** 41**,** "korhan"**,** 754334L**};**

cout **<<** get**<**AGE**>(**tp**)** **<<** " " **<<** get**<**NAME**(**tp**)** **<<** " " **<<** get**<**ID**>(**tp**)** **<<** "\n"**;**

**}**

**using** **namespace** std**;**

**using** PersonData **=** std**::**tuple**<int,** std**::**string**,** Date**>;**

PersonData get\_person\_data**()**

**{**

**return** **{** Irand**(**20**,** 60**)(),** rname**(),** Date**::**random**()** **}**

**}**

**int** main**()**

**{**

**for** **(int** i **=** 0 **;** i **<** 10**;** **++**i**)**

**{**

**auto** **[**age\_name**,** emp\_date**]** **=** get\_person\_data**();** ***// structed binding***

cout **<<** age **<<** " " **<<** name **<<** " " **<<** emp\_date**;**

**}**

**}**

***// structed binding***

**int** main**()**

**{**

**auto** tp **=** make\_tuple**(**235**,** 5.6**,** "emre"**);**

**auto** **[**x**,** y**,** z**]** **=** tp**;**

**}**

***// structed binding -- C dizileri***

**int** main**()**

**{**

**int** a**[**3**]**() **{** 54**,** 78**,** 90**};**

**auto** **&[**x**,** y**,** z**]** **=** a**;** ***// dizideki öğe sayısı ile structed bind öğe sayısı aynı olmalı***

cout **<<** "x = " **<<** x **<<** "\n"**;**

cout **<<** "y = " **<<** y **<<** "\n"**;**

cout **<<** "z = " **<<** z **<<** "\n"**;**

a**[**1**]** **\*=** 10**;**

cout **<<** "y ? " **<<** y **<<** "\n"**;** ***// y = 780***

**}**

***// structed binding --struct***

**struct** Data

**{**

**int** a**,** b**,** c**;**

**};**

Data foo**()**

**{**

**return** **{** 10**,** 20 **,**30 **};**

**}**

**int** main**()**

**{**

**auto** **[**a**,** b**,** \_**]** **=** foo**();**

**}**

**tuple\_size and tuple\_element**

**using** **namespace** std**;**

**using** ttype **=** std**::**tuple**<int,** **char,** **double,** Date**>;**

**int** main**()**

**{**

**consteval** **auto** n **=** tuple\_size**<**ttype**>::**value**;** ***// n = 4***

**consteval** **auto** n1 **=** tuple\_size\_v**<**ttype**>;** ***// n1 = 4***

tuple\_element**<**1**,** ttype**>::**type ***// char türü***

tuple\_element\_t**<**2**,** ttype**>** ***// dobule türü***

**}**

**std::tie --reference tutan tuple**

***// reference tutan tuple***

**int** main**()**

**{**

**int** a **=** 23**;** **double** dval **=** 4.02**;** string name **{**"emre"**};**

tuple**<int&,** **double&,** string**&>** tp**{** a**,** dval**,** name**};**

***// ya da***

**auto** tp1 **=** make\_tuple**(**ref**(**a**),** ref**(**dval**),** ref**(**name**));**

***// ya da***

**auto** tp2 **=** tie**(**a**,** dval**,** name**);** ***// reference tuple döndürür***

a **\*=** 100**;**

dval **+=**12.0**;**

name **+=** "gano"

cout **<<** get**<**0**>(**tp**)** **<<** '\n'**;**

cout **<<** get**<**1**>(**tp**)** **<<** '\n'**;**

cout **<<** get**<**2**>(**tp**)** **<<** '\n'**;**

**}**

tuple**<int,** **double,** string**>** foo**()**

**{**

**return** **{** 12**,** 4.56**,** "emre" **};**

**}**

**int** main**()**

**{**

**int** ival**;**

**double** dval**;**

string name**;**

tie**(**ival**,** dval**,** name**)** **==** foo**();**

**}**

**std::tuple karşılaştırma**

***// std::tuple karşılaştırma***

**using** **namespace** std**;**

***// id town name tarih***

**using** person **=** std**::**tuple**<int,** string**,** string**,** Date**>**

**int** main**()**

**{**

vector**<**person**>** pvec**;**

pvec**.**reserve**(**20'000**);**

**for** **(int** **=** 0**;** i **<** 20'000**;** **++**i**)**

**{**

pvec**.**emplace\_back**(**Irand**{**0**,** 100**},** rtown**(),** rname**(),** Date**::**random**());**

**}**

sort**(**pvec**.**begin**(),** pvec**.**end**());** ***// id -> town -> name -> tarih bu sıraya göre sıralar***

**}**

**class** Date

**{**

**public:**

Date**(int** day**,** **int** mon**,** **int** year**);**

**friend** **bool** **operator<const** Date**&** d1**,** **const** Date**&** d2**)**

**{**

**if** **(**d1**.**myear **!=** d2**.**myear**)**

**return** d1**.**myear **<<** d2**.**myear**;**

**if** **(**d1**.**mmon **!=** d2**.**mmon**)**

**return** d1**.**myear **<<** d2**.**myear**;**

**return** d1**.**mday **<** d2**.**mday**;**

***// yukarıdaki yerine***

**return** std**::**tuple**(**d1**.**myear**,** d1**.**mmon**,** d1**.**mday**)**

**<** std**::**tuple**(**d2**.**myear**,** d2**.**mmon**,** d2**.**mday**);**

**}**

**private:**

**int** mday**;**

**int** mmon**;**

**int** myear**;**

**};**

***// rotating assignment***

**int** main**()**

**{**

**int** x **=** 10**;** **int** y **=** 20**;** **int** z **=** 30**;** **int** t **=** 40**;**

***// x = 40, y = 10, z = 20, t = 10;***

**int** temp **=** x**;**

x **=** y**;**

y **=** z**;**

z **=** t**;**

t **=** temp**;**

***// ya da***

tie**(**x**,** y**,** z**,** t**)** **=** tuple**(**y**,** z**,** t**,** x**);** ***// x = 40, y = 10, z = 20, t = 10;***

**}**

**std::apply**

**using** **namespace** std**;**

**int** sum**(int** x**,** **int** y**,** **int** z**)**

**{**

**return** x **+** y **+** z**;**

**}**

**int** main**()**

**{**

tuple tx **{** 3**,** 6**,** 9**};**

**auto** ret **=** apply**(**sum**,** tx**);**

cout **<<** "ret = " **<<** ret **<<** "\n"**;**

**}**

**std::invoke**

***// std::invoke***

#include <functional>

**using** **namespace** std**;**

**int** sum**(int** x**,** **int** y**)**

**{**

**return** x **+** y**;**

**}**

**int** main**()**

**{**

**int** a **=** 5**,** **int** b **=** 9**;**

**int** ret **=** sum**(**a**,** b**);**

ret **=** invoke**(**sum**,** a**,** b**);** ***// generic kodlarda kullanılır***

**}**

**int** foo**(int);**

**int** main**()**

**{**

***// &foo int (\*)(int)***

***// foo int(int)***

**int** **(\***fp1**)(int)** **==** foo**;**

**int** **(\***fp2**)(int)** **==** **&**foo**;**

**}**

***// Member Functions Pointers***

**class** Nec

**{**

**public:**

**static** **int** foo**(int);**

**int** bar**(int);**

**};**

**int** main**()**

**{**

**int** **(\***fp**)(int)** **=** **&**Nec**::**foo**;**

**int** **(\***fp1**)(int)** **=** **&**Nec**::**bar**;** ***// syntax hatası***

**int** **(**Nec**::\***fp2**)(int)** **=** **&**Nec**::**foo**;**

**}**

**.\* operator**

**class** Nec

**{**

**public:**

**int** foo**(int)**

**{**

std**::**cout **<<** " Nec::foo(int x) x = " **<<** x **<<** "\n"**;**

**return** x **\*** x**;**

**}**

**};**

**int** main**()**

**{**

**auto** fp **=** **&**Nec**::**foo**;**

Nec mynec**;**

**(**mynec**.\***fp**)(**20**);**

**}**

**class** Nec

**{**

**public:**

**int** foo**(int** x**);**

**int** bar**(int** x**);**

**int** baz**(int** x**);**

**int** func**(int** x**);**

**};**

**using** necfp **=** **int** **(**Nec**::\*)(int);**

**int** main**()**

**{**

necfp fptr**;**

Nec mynec**;**

fptr **=** **&**Nec**::**foo**;**

fptr **=** **&**Nec**::**bar**;**

fptr **=** **&**Nec**::**baz**;**

**(**mynec**.\***fptr**)(**456**);**

vector**<**necfp**>** myvec**{** **&**Nec**::**foo**,** **&**Nec**::**bar**};**

myvec**.**push\_back**(&**Nec**::**baz**);**

myvec**.**push\_back**(&**Nec**::**func**);**

necfp ar**[]** **=** **{** **&**Nec**::**foo**,** **&**Nec**::**bar**,** **&**Nec**::**baz**,** **&**Nec**::**func**};**

**}**

**Bir fonksiyonun hangi member fonksiyonu çağıracağını seçicek**

**class** Nec

**{**

**public:**

**void** func**()**

**{**

**(this->\***mfp**)(**34**)**

**}**

**int** foo**(int** x**)**

**{**

std**::**cout **<<** "Nec::foo(int x) x = " **<<** x **<<** "\n"**;**

**return** x **+** 5**;**

**}**

**int** bar**(int** x**)**

**{**

std**::**cout **<<** "Nec::bar(int x) x = " **<<** x **<<** "\n"**;**

**return** x **\*** x**;**

**}**

**int** baz**(int** x**)**

**{**

std**::**cout **<<** "Nec::baz(int x) x = " **<<** x **<<** "\n"**;**

**return** x **\*** x **\*** xx**;**

**}**

**private:**

**int** **(**Nec**::\***mfp**)(int)** **=** **&**Nec**::**foo**;**

**};**

**void** gf**(**Nec**&** nec**,** **int** **(**Nec**::\***fp**)(int))**

**{**

**(**nec**.\***fp**)(**21**);**

**}**

**int** main**()**

**{**

Nec mynec**;**

gf**(**mynec**,** **&**Nec**::**bar**);**

**}**

**int** main**()**

**{**

**int** **(**Nec**::** **\***fp**)(int)** **=** **&**Nec**::**bar**;**

Nec**\*** pnec **=** **new** Nec**;**

**((\***pnec**).\***fp**)(**345**);**

***// ya da***

**(**pnec**->\***fp**)(**457**);**

***// invoke***

std**::**invoke**(**fp**,** mynec**,** 345**);**

std**::**invoke**(**fp**,** necptr**,** 99**);**

**}**