Сложни типове данни

Увод в паралелното програмиране #7

Материалите са разработени в рамките на договор ДО-02-136/2008 с Фонд Научни Изследвания
Проект: IRC-CoSiM Ръководител на проекта: проф. дфзн Ана Пройкова, ФзФ на СУ
http://www.irc-cosim.uni-sofia.bg/

Общи положения

- Стандартните предефинирани типове на MPI са скаларни и представляват един елемент от примитивен за езика тип
- Със стандартните типове могат да се предава само съдържанието на непрекъснати области в паметта (масиви)
- Структурите на C и COMMON блоковете на Fortran не трябва да се предават като набор байтове от съображения за преносимост и работа в хетерогенна среда
- Достъп до съдържанието на паметта по по-сложна схема
- Произволно влагане на сложните типове

Карта на типа

- Всеки тип в MPI има асоциирана карта на типа

  { (*type*$_0$, *displ*$_0$), (*type*$_1$, *displ*$_1$), ..., (*type*$_{N-1}$, *displ*$_{N-1}$) }
  N - брой на примитивните типове в MPI типа

- *type*$_0$, *type*$_1$ и т.н. са примитивни за езика типове данни
- *displ*$_0$, *displ*$_1$ и т.н. са отместванията в байтове от началото на адреса в паметта и могат да бъдат отрицателни
- Картата се използва при пакетиране и разпакетиране на MPI съобщенията
- Простите типове имат само по един елемент на отместване 0:
  - *MPI_INT* – { (int, 0) }
  - *MPI_LOGICAL* – { (LOGICAL, 0) }

Обхват и размер на типа

- *Долната граница* на типа е отместването на първия (според наредбата в паметта) елемент
- *Горната граница* на типа е отместването след края на последния (според наредбата в паметта) елемент
- *Обхватът* на типа е разликата между горната и долната му граница
- *Размерът* на типа е броят на байтовете в пакетираното съдържание
- Типът { *(int, 0), (byte, -4), (byte, 30)* } има:
  - долна граница: -4 байта
  - горна граница: 31 байта
  - обхват: 35 байта
  - размер: 6 байта (ако int е 4 байта)

Обхват и размер на типа

- *MPI_Type_lb (datatype, displ)*
  MPI_Type_ub (datatype, displ)
  - връщат съответната граница на регистрирания тип *datatype*
- *MPI_Type_extent (datatype, extent)*
  - връща обхвата на регистрирания тип *datatype*
  - MPI-2 препоръчва използването на *MPI_Type_get_extent*, което връща и долната граница на типа
- *MPI_Type_size (datatype, size)*
  - връща размера на регистрирания тип *datatype*

Сложни типове

- Състоят се един или повече елементи от предварително съществуващ MPI тип
- Четири вида сложни типове:
  - последователни елементи
  - “разкрачени” вектори
  - индексирани типове
  - структури
- Сложните типове се регистрират с *MPI_Type_commit()* и се освобождават с *MPI_Type_free()*

Последователни елементи

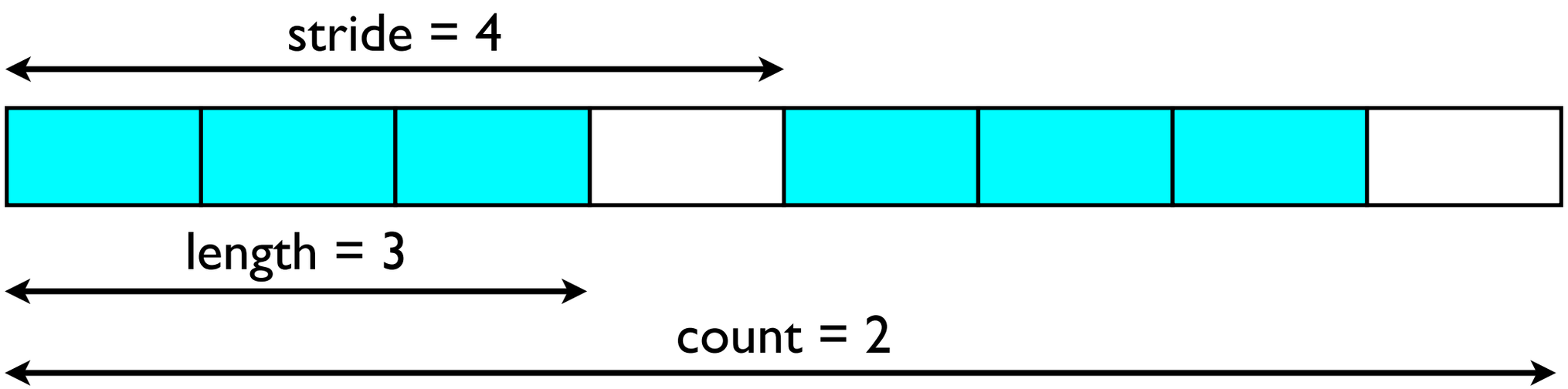
MPI_Type_contiguous (count, oldtype, newtype)

- Създава тип, обхващащ *count* на брой последователни елемента от тип *oldtype*
- Разстоянието в паметта между последователните елементи е равно на обхвата на *oldtype*
- Връща манипулатор на новия тип в *newtype*
- *newtype* трябва да се регистрира с извикване на *MPI_Type_commit()*
- Удобен за предаване на цели редове (C) или колони (Fortran) на матрица

“Разкрачени” вектори

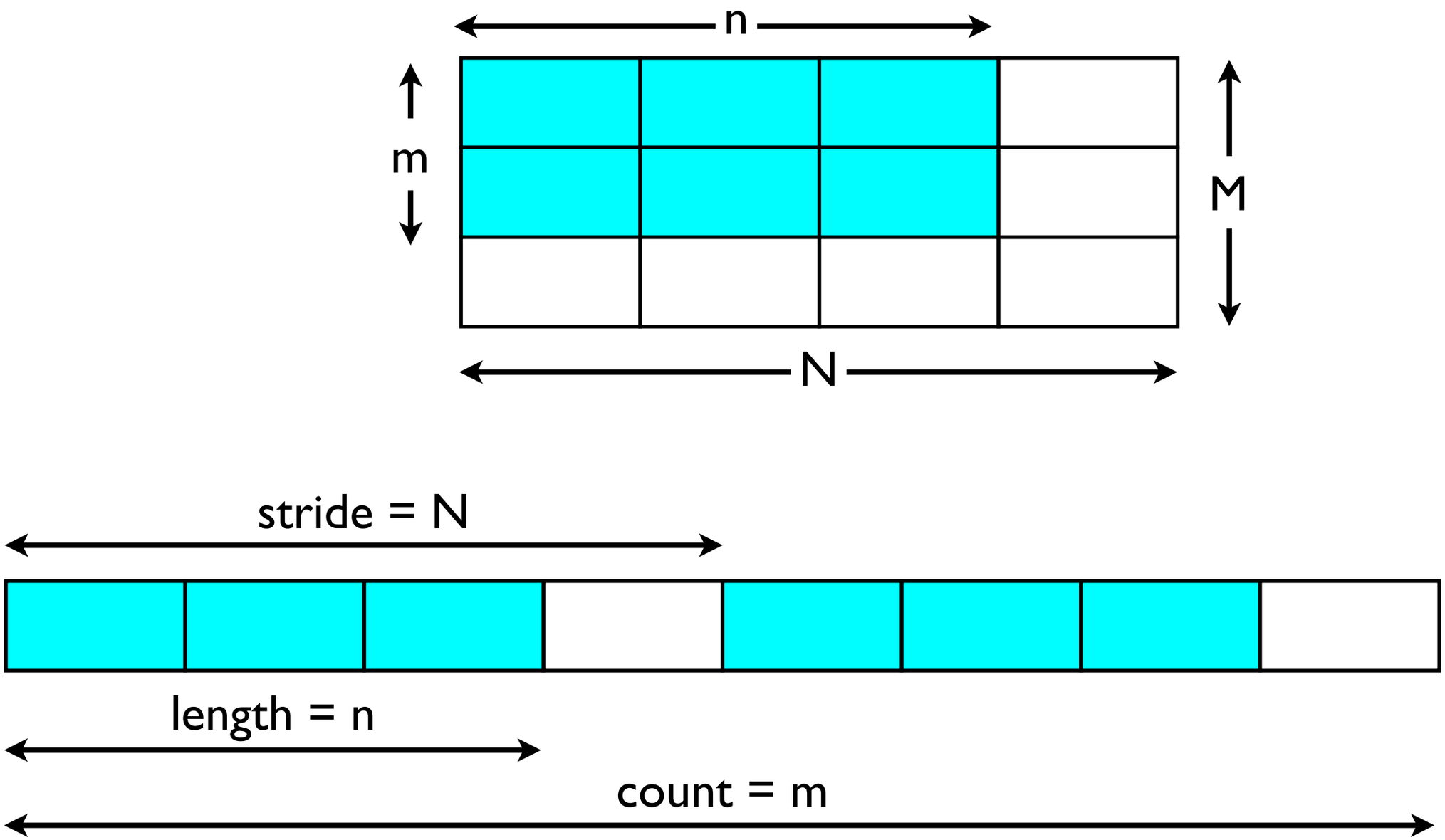
MPI_Type_vector (count, length, stride, oldtype, newtype)

- Създава тип, обхващащ *count* на брой последователности от по *length* елемента от тип *oldtype*, отстоящи на *stride* елемента една от друга



- Удобен за предаване на подматрици

Подматрици

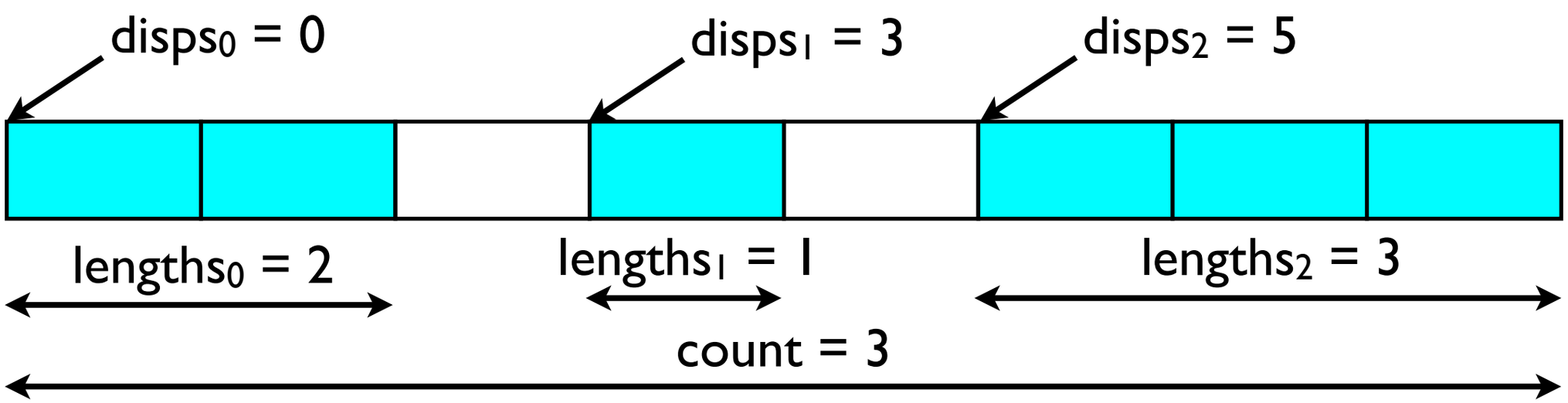


Забележка: във Fortran наредбата на елементите е по колони и там:
stride = M, *length* = m, *count* = n

Индексирани типове

MPI_Type_indexed (count, lengths, disps, oldtype, newtype)

- Създава тип, обхващащ *count* на брой последователности от по $lengths_i$ елемента от тип *oldtype*, отстоящи на $disps_i$ елемента от началото



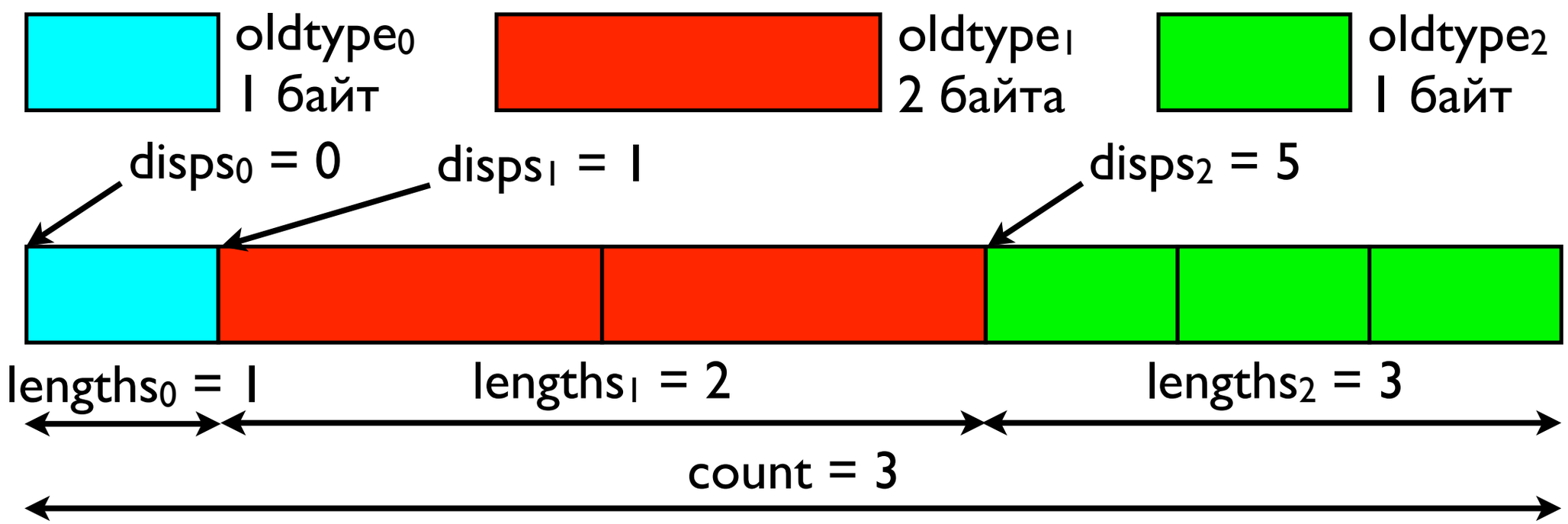
“Байтови” версии

- *MPI_Type_hvector()*:
  - заменя големината на разкрача в брой елементи с големина на разкрача в байтове
  - MPI-2 препоръчва използването на *MPI_Type_create_hvector()*
- *MPI_Type_hindexed()*:
  - заменя отместванията в брой елементи спрямо началото с отмествания в байтове спрямо началото
  - MPI-2 препоръчва използването на *MPI_Type_create_hindexed()*

Структури

MPI_Type_struct (count, lengths, disps, oldtypes, newtype)

- Създава тип, обхващащ *count* на брой последователности от по $lengths_i$ елемента от тип $oldtype_i$, отстоящи на $disps_i$ байта от началото



- MPI-2 препоръчва използването на *MPI_Type_create_struct()*

Структури - пример от C

```
struct atom
{
  int type;
  double r[3];
  double v[3];
  double f[3];
};
```

oldtypes[0] = MPI_INT
lengths[0] = 1
disps[0] = offsetof(struct atom, type)

oldtypes[1] = MPI_DOUBLE
lengths[1] = 3
disps[1] = offsetof(struct atom, r)

oldtypes[2] = MPI_DOUBLE
lengths[2] = 3
disps[2] = offsetof(struct atom, v)

oldtypes[3] = MPI_DOUBLE
lengths[3] = 3
disps[3] = offsetof(struct atom, f)

Структури в C

- Възможно е елементите да не следват плътно един след друг – padding
  - хардуерни ограничения или от съображения за скорост
- Препоръчва се използването на *offsetof()* вместо събиране на *sizeof()* на отделните членове
- При предаване на масиви от структури е добре в MPI типа да се дефинира още един елемент от тип *MPI_UB* на отместване sizeof(структурата):

```
oldtypes[n] = MPI_UB
lengths[n] = 1
disps[n] = sizeof(struct atom)
```

Модификация на обхвата на структурата

- MPI позволява употребата на псевдотиповете *MPI_LB* и *MPI_UB*, с които може да се маркират границите на типа на структурата
- *MPI_LB* и *MPI_UB* имат нулев размер и обхват и не заемат място в съобщението
- Минималното отместване на елемент от тип *MPI_LB* става долна граница на типа
- Максималното отместване на елемент от тип *MPI_UB* става горна граница на типа
- Размерът на типа се запазва

Структури във Fortran

- MPI структурите могат да се използват за предаване на съдържанието на COMMON блокове
- Адресите на отделните променливи в блока могат да се получат с *MPI_Get_address (var, addr)*
- Масиви от COMMON блокове по принцип няма и не се налага модифицирането на горната граница на типа

Съобщения от сложни типове

- Могат да се съвместяват сложни типове и масиви от базовия тип от двете страни на операцията по предаване на съобщение
- Единствено изискване при получаване е размерът на сложния тип да дели размера на съобщението
- В противен случай се получават нецял брой елементи от сложния тип и *MPI_Get_count()* връща *MPI_UNDEFINED*
- За сложните типове с един базов тип може да се използва *MPI_Get_elements()*