# CS 2211
# Systems Programming

## Part Ten – A

## Structures and Memory

## POINTERS

### Passing Values TO and FROM a Function

```
void swap_value(int va, int vb) {
  int vTmp = va;
  va = vb;
  vb = vTmp;
}

void swap_reference(int *ra, int *rb) {
  int rTmp = *ra;
  *ra = *rb;
  *rb = rTmp;
}

int main()
{
    int a = 1;
    int b = 2;
    printf("before swaps: a = %d\n", a);
    printf("before swaps: b = %d\n", b);
    swap_value(a, b);
    printf("after swap_value: a = %d\n", a);
    printf("after swap_value: b = %d\n", b);

    swap_reference(&a, &b);
    printf("after swap_reference: a = %d\n", a);
    printf("after swap_reference: b = %d\n", b);
}
```

pass-by-value
verus
pass-by-reference

# POINTERS

( 1, 2 )

## Passing Values TO and FROM a Function

```
void swap_value(int va, int vb) {
  int vTmp = va;
  va = vb;
  vb = vTmp;
}

void swap_reference(int *ra, int *rb) {
  int rTmp = *ra;
  *ra = *rb;
  *rb = rTmp;
}

int main()
{
    int a = 1;
    int b = 2;
    printf("before swaps: a = %d\n", a);
    printf("before swaps: b = %d\n", b);
    swap_value(a, b);
    printf("after swap_value: a = %d\n", a);
    printf("after swap_value: b = %d\n", b);

    swap_reference(&a, &b);
    printf("after swap_reference: a = %d\n", a);
    printf("after swap_reference: b = %d\n", b);
}
```

call frame (main)

| Label | Address | Value |
|---|---|---|
| a | 400 - 403 | 1 |
| b | 404 - 407 | 2 |
| va | 512 - 515 | 1 |
| vb | 516 -519 | 2 |
| | | |
| | | |
| | | |
| | | |
| | | |

## POINTERS

## Passing Values TO and FROM a Function

```
void swap_value(int va, int vb) {
  int vTmp = va;
  va = vb;
  vb = vTmp;
}

void swap_reference(int *ra, int *rb) {
  int rTmp = *ra;
  *ra = *rb;
  *rb = rTmp;
}

int main()
{
    int a = 1;
    int b = 2;
    printf("before swaps: a = %d\n", a);
    printf("before swaps: b = %d\n", b);
    swap_value(a, b);
    printf("after swap_value: a = %d\n", a);
    printf("after swap_value: b = %d\n", b);

    swap_reference(&a, &b);
    printf("after swap_reference: a = %d\n", a);
    printf("after swap_reference: b = %d\n", b);
}
```

( 400, 404 )

call frame (main)

| Label | Address | Value |
|---|---|---|
| a | 400 - 403 | 1 |
| b | 404 - 407 | 2 |
| ra | 512 - 515 | 400 |
| rb | 516 -519 | 404 |
| | | |
| | | |
| | | |
| | | |
| | | |

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  teacher;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   DisplayStats(teacher);
   printf("%s\n",teacher.first);


   return 0;
}
```

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;
        } person;

int main()
{

   person  teacher;

   teacher.year=2025;
   strcpy(teacher.firs
   strcpy(teacher.last

   DisplayStats(teache
   printf("%s\n",teach


   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first[0] | 400 - | S |
| | teacher.first[1] | 401 - | a |
| | teacher.first[2] | 402 - | m |
| | teacher.first[3] | 403 - | \0 |
| | teacher.first[5] | 404 - | |
| | teacher.first[6] – [31] | 405 - 431 | |
| teacher.last | teacher.last[0] | 432 - | M |
| | teacher.last[1] | 433 - | a |
| | teacher.last[2] | 434 - | g |
| | teacher.last[3] | 435 - | g |
| | teacher.last[4] | 436 - | s |
| | teacher.last[5] | 437 - | \0 |
| | teacher.last[5] | 438 - | |
| | teacher.last[6] – [31] | 439 - 463 | |
| | teacher.year | 464 - 467 | 2025 |
| | | | |
| | | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  teacher;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   DisplayStats(teacher);
   printf("%s\n",teacher.first);


   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| | | | |
| | | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{
   person  teacher, student;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   student = teacher;
   PrintStructure(student);

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);

   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | |
| teacher.last | teacher.last | 432 - 463 | |
| | teacher.year | 464 - 467 | |
| student<br>student.first | student.first | 468 - 499 | |
| student.last | student.last | 500 - 531 | |
| | student.year | 532 - 535 | |
| | | | |
| | | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  teacher, student;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   student = teacher;
   PrintStructure(student);

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);

   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| student<br>student.first | student.first | 468 - 499 | |
| student.last | student.last | 500 - 531 | |
| | student.year | 532 - 535 | |
| | | | |
| | | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  teacher, student;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   student = teacher;
   PrintStructure(student);

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);

   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| student<br>student.first | student.first | 468 - 499 | Sam |
| student.last | student.last | 500 - 531 | Maggs |
| | student.year | 532 - 535 | 2025 |
| | | | |
| | | | |

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  teacher;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);

   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| | | | |
| | | | |

Teacher


## POINTERS

## Passing Values TO and FROM a Function

```
void UnChangeStruct( person inputStr)
{
   strcpy(Input.first,"Ima");
   strcpy(Input.last,"Dunsl");
   PrintStructure(inputStr);
}
```

a copy of input person.

```
ty                                          char */
                                            char */

                                            */
int main()
{

   person  teacher;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);
   return 0;
}
```

call frame (main)

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| inputStr<br>inputStr.first | inputStr.first | 620 - 651 | Ima |
| inputStr.last | inputStr.last | 652 - 683 | Dunsl |
| | inputStr.year | 684 - 687 | |
| | | | |
| | | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  teacher;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);

   // student = teacher;
   // PrintStructure(student);

   return 0;
}
```

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| | | | |
| | | | |

address of structure teacher.

# POINTERS

## Passing Values TO and FROM a Function

```
void ChangeStruct( person *refInputStr)
{
   strcpy(refInputStr.first,"Ima");
   strcpy(refInputStr.last,"Dunsl");
   refInputStr.year = 2042;
   PrintStructure(refInputStr);
}
```

```
ty                                          char */
                                            char */

                                            */
in
{
   person  teacher;

   teacher.year=2025;
   strcpy(teacher.first,"Sam");
   strcpy(teacher.last,"Maggs");

   UnChangeStruct(teacher);
   ChangeStruct(&teacher);
   PrintStructure(teacher);

   // student = teacher;
   // PrintStructure(student);

   return 0;
}
```

call frame (main)

| Address Label | Label | Address | Value |
|---|---|---|---|
| teacher<br>teacher.first | teacher.first | 400 - 431 | Sam |
| teacher.last | teacher.last | 432 - 463 | Maggs |
| | teacher.year | 464 - 467 | 2025 |
| | refInputStr | 640 - 643 | 400 |
| | | | |
| | | | |
| | | | |
| | | | |

# Structures and Memory in C

**END OF PART 1**

## POINTERS

### DYNAMIC MEMORY ALLOCATION

- **static memory allocation** (non-changing)
  - the size (in bytes) is known BEFORE
    a program starts to execute
  - when the program is loaded into memory,
    allocation of declared variables is performed

靜態

- sometimes a program does not know exactly
  how much memory it may need

i.e. reading a line of text – could be a character array of any size
- always declaring a humongous array very wasteful

- so: use **dynamic memory allocation**
  ask the O/S to set aside **x** amount of memory during execution

## POINTERS

### DYNAMIC MEMORY ALLOCATION

variations on the malloc() function

notice **malloc()** requests exact number of bytes

- programmer must know how many based on how to be used

**calloc()** - sets aside cells (memory) and initializes all to zero

```
double *a;

a = (double *) calloc (70, 8);
```

70 x 8 bytes
70 double variables
560 byes

```
// could have used
a = (double *) calloc ( 70, sizeof(double) );
```

- same thing -> sizeof(?) returns size of variable type based on O/S.

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
```

requesting O/S to set aside 40 bytes configured to handle values of type **double** and assign the address of this memory block in the pointer variable **a**

**NOTE: {DM}** – is not a label

- just something to put in temporary symbol for **allocated dynamic memory** (reserved memory for use later….)

| Label | Address | Value |
|---|---|---|
| | | |
| a | 400 - 403 | 10000 |
| | | |
| {DM} | 10000 - 10039 | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  *pTeacher;

   pTeacher = (person *)calloc(1, sizeof(person));

   pTeacher->year=2025;
   strcpy(pTeacher->first,"Sam");
   strcpy(pTeacher->last,"Maggs");

   UnChangeStruct(*teacher);
   ChangeStruct(teacher);
   PrintStructure(teacher);


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| {DM} | 10000 - 10067 | |
| | | |
| | | |
| | | |
| | | |
| | | |

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  *pTeacher;

   pTeacher = (person *)calloc(1, sizeof(person));

   pTeacher->year=2025;
   strcpy(pTeacher->first,"Sam");
   strcpy(pTeacher->last,"Maggs");

   UnChangeStruct(*teacher);
   ChangeStruct(teacher);
   PrintStructure(teacher);


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| pTeacher->first | 10000 - 10031 | Sam |
| pTeacher->last | 10032 - 10063 | Maggs |
| pTeacher->year | 10064 - 10067 | 2025 |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
    char first[32];
    char last[32];
    int year;
} person;

void UnChangeStruct( person inputStr)
{
    strcpy(inputstr.first,"Ima");
    strcpy(inputstr.last,"Dunsl");
    inputstr.year = 2022;
    PrintStructure(inputStr);
}

int main(void)
{
    person  *pTeacher;

    pTeacher = (person *)calloc(1, sizeof(person));

    pTeacher->year=2025;
    strcpy(pTeacher->first,"Sam");
    strcpy(pTeacher->last,"Maggs");

    UnChangeStruct(*pTeacher);
    ChangeStruct(pTeacher);
    PrintStructure(pTeacher);

    return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| inputStr.first | 640 - 671 | Ima |
| inputStr.last | 672 - 703 | Dunsl |
| inputStr.year | 704 - 707 | 2022 |
| | | |
| pTeacher->first | 10000 - 10031 | Sam |
| pTeacher->last | 10032 - 10063 | Maggs |
| pTeacher->year | 10064 - 10067 | 2025 |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void ChangeStruct( person *refInputStr)
{
   strcpy(refInputStr->first,"Ima");
   strcpy(refInputStr->last,"Dunsl");
   // refInputStr->year = 2042;
   PrintStructure(refInputStr);
}
```

```
ty
                                          char */
                                          char */

                                          */
in
{
   person  *pTeacher;

   pTeacher = (person *)calloc(1, size

   pTeacher->year=2025;
   strcpy(pTeacher->first,"Sam");
   strcpy(pTeacher->last,"Maggs");

   UnChangeStruct(*pTeacher);
   ChangeStruct(pTeacher);
   PrintStructure(pTeacher);


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| refInputStr | 500 - 503 | 10000 |
| | | |
| pTeacher->first | 10000 - 10031 | Sam |
| pTeacher->last | 10032 - 10063 | Maggs |
| pTeacher->year | 10064 - 10067 | 2025 |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void ChangeStruct( person *refInputStr)
{
   strcpy(refInputStr->first,"Ima");
   strcpy(refInputStr->last,"Dunsl");
   // refInputStr->year = 2042;
   PrintStructure(refInputStr);
}
```

```
ty
                                          char */
                                          char */

                                          */
in
{
   person  *pTeacher;

   pTeacher = (person *)calloc(1, size

   pTeacher->year=2025;
   strcpy(pTeacher->first,"Sam");
   strcpy(pTeacher->last,"Maggs");

   UnChangeStruct(*pTeacher);
   ChangeStruct(pTeacher);
   PrintStructure(pTeacher);


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| refInputStr | 500 - 503 | 10000 |
| | | |
| pTeacher->first | 10000 - 10031 | Ima |
| pTeacher->last | 10032 - 10063 | Dunsl |
| pTeacher->year | 10064 - 10067 | 2025 |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  *pTeacher;

   pTeacher = (person *)calloc(1, size

   pTeacher->year=2025;
   strcpy(pTeacher->first,"Sam");
   strcpy(pTeacher->last,"Maggs");

   UnChangeStruct(*pTeacher);
   ChangeStruct(pTeacher);
   PrintStructure(pTeacher);


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| pTeacher->first | 10000 - 10031 | Ima |
| pTeacher->last | 10032 - 10063 | Dunsl |
| pTeacher->year | 10064 - 10067 | 2025 |
| | | |
| | | |

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  *pTeacher;

   pTeacher = (person *)calloc(4, sizeof(person));

   pTeacher[2].year=2025;
   strcpy(pTeacher[2].first,"Sam");
   strcpy(pTeacher[2].last,"Maggs");

   UnChangeStruct(pTeacher[2]);
   ChangeStruct(&pTeacher[2]);
   PrintStructure(pTeacher[2]););


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| {DM} | 10000 - 10271 | |
| | | |
| | | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{

   person  *pTeacher;

   pTeacher = (person *)calloc(4, si

   pTeacher[2].year=2025;
   strcpy(pTeacher[2].first,"Sam");
   strcpy(pTeacher[2].last,"Maggs");

   UnChangeStruct(pTeacher[2]);
   ChangeStruct(&pTeacher[2]);
   PrintStructure(pTeacher[2]););


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| {DM} | 10000 - 10135 | |
| pTeacher[2].first | 10136 - 10167 | Sam |
| pTeacher[2].last | 10168 - 10199 | Maggs |
| pTeacher[2].year | 10200 - 10203 | 2025 |
| {DM} | 10204 - 10271 | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void UnChangeStruct( person inputStr)
{
    strcpy(inputstr.first,"Ima");
    strcpy(inputstr.last,"Dunsl");
    inputstr.year = 2022;
    PrintStructure(inputStr);
}
```

```
typedef struct human {
                                                    har */
                                                    har */
                                                    /
int
{
    person  *pTeacher;

    pTeacher = (person *)calloc(4, si

    pTeacher[2].year=2025;
    strcpy(pTeacher[2].first,"Sam");
    strcpy(pTeacher[2].last,"Maggs");

    UnChangeStruct(pTeacher[2]);
    ChangeStruct(&pTeacher[2]);
    PrintStructure(pTeacher[2]););


    return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| inputStr.first | 640 - 671 | Ima |
| inputStr.last | 672 - 703 | Dunsl |
| inputStr.year | 704 - 707 | 2022 |
| | | |
| {DM} | 10000 - 10135 | |
| pTeacher[2].first | 10136 - 10167 | Sam |
| pTeacher[2].last | 10168 - 10199 | Maggs |
| pTeacher[2].year | 10200 - 10203 | 2025 |
| {DM} | 10204 - 10271 | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void ChangeStruct( person *refInputStr)
{
   strcpy(refInputStr->first,"Ima");
   strcpy(refInputStr->last,"Dunsl");
   // refInputStr->year = 2042;
   PrintStructure(refInputStr);
}
```

```
ty
                                          char */
                                          char */

                                          */
in
{
   person  *pTeacher;

   pTeacher = (person *)calloc(4, si

   pTeacher[2].year=2025;
   strcpy(pTeacher[2].first,"Sam");
   strcpy(pTeacher[2].last,"Maggs");

   UnChangeStruct(pTeacher[2]);
   ChangeStruct(&pTeacher[2]);
   PrintStructure(pTeacher[2]););


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| refInputStr | 500 - 503 | 10136 |
| | | |
| {DM} | 10000 - 10135 | |
| pTeacher[2].first | 10136 - 10167 | Sam |
| pTeacher[2].last | 10168 - 10199 | Maggs |
| pTeacher[2].year | 10200 - 10203 | 2025 |
| {DM} | 10204 - 10271 | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void ChangeStruct( person *refInputStr)
{
   strcpy(refInputStr->first,"Ima");
   strcpy(refInputStr->last,"Dunsl");
   // refInputStr->year = 2042;
   PrintStructure(refInputStr);
}
```

```
ty
                                        char */
                                        char */

                                        */
in
{
   person  *pTeacher;

   pTeacher = (person *)calloc(4, si

   pTeacher[2].year=2025;
   strcpy(pTeacher[2].first,"Sam");
   strcpy(pTeacher[2].last,"Maggs");

   UnChangeStruct(pTeacher[2]);
   ChangeStruct(&pTeacher[2]);
   PrintStructure(pTeacher[2]););


   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| pTeacher | 400 - 403 | 10000 |
| | | |
| refInputStr | 500 - 503 | 10136 |
| | | |
| {DM} | 10000 - 10135 | |
| pTeacher[2].first | 10136 - 10167 | Ima |
| pTeacher[2].last | 10168 - 10199 | Dunsl |
| pTeacher[2].year | 10200 - 10203 | 2025 |
| {DM} | 10204 - 10271 | |
| | | |
| | | |

# Structures and Memory in C

**END OF PART 2**

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);

/* can just assume the malloc(40) call set aside
   an area in the heap to accommodate 5 double variables
   (40 bytes is:
      8 bytes (size of a double variable) x 5 )
   so, the virtual labels immediately available for use )
```

| Label | | Address | Value |
|---|---|---|---|
| | | | |
| | a | 400 - 403 | 10000 |
| | | | |
| *(a+0) | a[0] | 10000 - 10007 | |
| *(a+1) | a[1] | 10008 - 10015 | |
| *(a+2) | a[2] | 10016 - 10023 | |
| *(a+3) | a[3] | 10024 - 10031 | |
| *(a+4) | a[4] | 10032 - 10039 | |
| | | | |

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
A[0] = 37;
A[2] = 64;
...
// BUT! What if five elements was not enough !
```

| Label | | Address | Value |
|---|---|---|---|
| | | | |
| | a | 400 - 403 | 10000 |
| | | | |
| *(a+0) | a[0] | 10000 - 10007 | 37 |
| *(a+1) | a[1] | 10008 - 10015 | |
| *(a+2) | a[2] | 10016 - 10023 | 64 |
| *(a+3) | a[3] | 10024 - 10031 | |
| *(a+4) | a[4] | 10032 - 10039 | |
| | | | |

## POINTERS

### DYNAMIC MEMORY ALLOCATION

variations on the malloc() function

notice **malloc()** requests exact number of bytes
- programmer must know how many based on how to be used

**realloc()** - is used to resize allocated memory without losing existing data

```
double *a;
a = (double *)malloc(5 *sizeof(double));

a = (double *) realloc (a, 9*sizeof(double));
```

9 x 8 bytes
increase to 9 double variables
72 byes

```
// could have used
a = (double *) realloc (a, 3*sizeof(double));
```

- decrease to 3 double variables
24 bytes.

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
A[0] = 37;
A[2] = 64;
...
// BUT! What if five elements was not enough !
```

| Label | Address | Value |
|---|---|---|
| | | |
| a | 400 - 403 | 10000 |
| | | |
| {DM} | 10000 - 10039 | |
| | | |
| | | |
| | | |
| | | |
| | | |

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
A[0] = 37;
A[2] = 64;
...
// BUT! What if five elements was not enough !
```

| Label | | Address | Value |
|---|---|---|---|
| | | | |
| | a | 400 - 403 | 10000 |
| | | | |
| *(a+0) | a[0] | 10000 - 10007 | 37 |
| *(a+1) | a[1] | 10008 - 10015 | |
| *(a+2) | a[2] | 10016 - 10023 | 64 |
| *(a+3) | a[3] | 10024 - 10031 | |
| *(a+4) | a[4] | 10032 - 10039 | |
| | | | |

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
A[0] = 37;
A[2] = 64;
...
// BUT! What if five elements was not enough !
a = realloc ( a, 7 * sizeof(double) );
```

| Label | Address | Value |
|---|---|---|
| | | |
| a | 400 - 403 | 10000 |
| | | |
| {DM} | 10000 - 10039 | |
| | | |
| | | |
| | | |
| | | |
| | | |

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
A[0] = 37;
A[2] = 64;
...
// BUT! What if five elements was not enough !
a = realloc ( a, 7 * sizeof(double) );
```

| Label | Address | Value |
|---|---|---|
| | | |
| a | 400 - 403 | 10000 |
| | | |
| {DM} | 10000 - 10055 | |
| | | |
| | | |
| | | |
| | | |
| | | |

## POINTERS

```
double *a;      /* a pointer variable */
a = ( double *) malloc (40);
A[0] = 37;
A[2] = 64;
...
// BUT! What if five elements was not enough !
a = realloc ( a, 7 * sizeof(double) );
```

| Label | | Address | Value |
|---|---|---|---|
| | | | |
| | a | 400 - 403 | 10000 |
| | | | |
| *(a+0) | a[0] | 10000 - 10007 | 37 |
| *(a+1) | a[1] | 10008 - 10015 | |
| *(a+2) | a[2] | 10016 - 10023 | 64 |
| *(a+3) | a[3] | 10024 - 10031 | |
| *(a+4) | a[4] | 10032 - 10039 | |
| *(a+5) | a[5] | 10040 - 10047 | |
| *(a+6) | a[6] | 10048 - 10055 | |
| | | | |

## POINTERS

```
dou
a = ( double *) malloc (40);
A[0] =
A[2] =
...
// BUT!
a = rea
```

1st case: If sufficient memory is available after address 10039, then the address of **a** doesn't change.

2nd case: If sufficient memory is not available after address 10039, then the realloc() function allocates memory somewhere else in the heap and copies the all content from old memory block to the new memory block.
In this case the address of **a** changes.

| Label | | Address | Value |
|---|---|---|---|
| | | | |
| | a | 400 - 403 | 10000 |
| | | | |
| *(a+0) | a[0] | 10000 - 10007 | 37 |
| *(a+1) | a[1] | 10008 - 10015 | |
| *(a+2) | a[2] | 10016 - 10023 | 64 |
| *(a+3) | a[3] | 10024 - 10031 | |
| *(a+4) | a[4] | 10032 - 10039 | |
| *(a+5) | a[5] | 10040 - 10047 | |
| *(a+6) | a[6] | 10048 - 10055 | |
| | | | |

# Structures and Memory in C

**END OF PART 3**

## POINTERS (double pointers)

```
int i;          /* an integer variable */
int *ptr1_i;    /* a pointer variable that points to i */
int **ptr2_i;   /* a pointer variable that points to i */
int x;          /* another integer variable */

i = 37;
ptr1_i = &i;
ptr2_i = &ptr1_i;
x = **ptr2_i;
```

| Label | Address | Value |
|---|---|---|
| | | |
| i | 400 - 403 | 37 |
| ptr1_i | 404 - 407 | 400 |
| ptr2_i | 408 - 411 | 404 |
| x | 412 - 415 | 37 |
| | | |
| | | |
| | | |
| | | |

```
#include <stdio.h>

int twoDimArray(int **passedArray)
{
    printf("address of passedArray[0]: %u\n", &passedArray[0]);
    printf("address of passedArray[1]: %u\n", &passedArray[1]);
    passedArray [0][1] = 56;
    *(*(passedArray+1)+0) = -31;    /* same as passedArray [1][0] */
}


int main(int argc, char *argv[])
{
    int **m;                    /* 4 bytes (just an address) */
    m = (int **) calloc (2, sizeof(int *) ); /* 2 x 4 bytes */
    m[0] = (int *) calloc ( 3, sizeof(int)); /* 3 x 4 bytes */
    m[1] = (int *) calloc ( 2, sizeof(int)); /* 2 x 4 bytes */
    printf("address of m[0]: %u\n", &m[0]);
    printf("address of m[1]: %u\n", &m[1]);

    twoDimArray(m);

    printf("value of m[0][1]: %d \n",m[0][1]);
    printf("value of m[1][0]: %d \n",m[1][0]);

}
```

## POINTERS (double pointers)

```
int **m;                          /* 4 bytes (just an address) */
m = (int **) calloc (2, sizeof(int *) ); /* 2 x 4 bytes */
m[0] = (int *) calloc ( 3, sizeof(int)); /* 3 x 4 bytes */
m[1] = (int *) calloc ( 2, sizeof(int)); /* 2 x 4 bytes */
m[0][1] = 56;
*(*(m+1)+0) = -31;      /* same as m[1][0] – either okay */
```

| Label | Address | Value |
|---|---|---|
| m | 400 - 404 | 10100 |
| | | |
| {DM} | 10100 - 10107 | |
| {DM} | 10108 - 10119 | |
| {DM} | 10120 - 10127 | |
| | | |
| | | |
| | | |
| | | |
| | | |

| Label | | Address | Value |
|---|---|---|---|
| m | | 400 - 404 | 10100 |
| | | | |
| *(m+0) | m[0] | 10100 - 10103 | 10108 |
| *(m+1) | m[1] | 10104 - 10107 | 10120 |
| *(*(m+0)+0) | m[0][0] | 10108 - 10111 | |
| *(*(m+0)+1) | m[0][1] | 10112 - 10115 | 56 |
| *(*(m+0)+2) | m[0][2] | 10116 - 10119 | |
| *(*(m+1)+0) | m[1][0] | 10120 - 10123 | -31 |
| *(*(m+1)+1) | m[1][1] | 10124 - 10127 | |
| | | | |

```
#include <stdio.h>

int twoDimArray(int **passedArray)
{
    printf("address of passedArray[0]: %u\n", &passedArray[0]);
    printf("address of passedArray[1]: %u\n", &passedArray[1]);
    passedArray [0][1] = 56;
    *(*(passedArray+1)+0) = -31;    /* same as passedArray [1][0] */
}


int main(int argc, char *argv[])
{
    int **m;
    m = (int **) calloc (2,
    m[0] = (int *) calloc (
    m[1] = (int *) calloc (
    printf("address of m[0]:
    printf("address of m[1]:

    twoDimArray(m);

    printf("value of m[0][1]
    printf("value of m[1][0]

}
```

| Label | Address | Value |
|---|---|---|
| m | 400 - 404 | 10100 |
| | | |
| *(m+0) m[0] | 10100 - 10103 | 10108 |
| *(m+1) m[1] | 10104 - 10107 | 10120 |
| *(*(m+0)+0) m[0][0] | 10108 - 10111 | |
| *(*(m+0)+1) m[0][1] | 10112 - 10115 | 56 |
| *(*(m+0)+2) m[0][2] | 10116 - 10119 | |
| *(*(m+1)+0) m[1][0] | 10120 - 10123 | -31 |
| *(*(m+1)+1) m[1][1] | 10124 - 10127 | |
| | | |

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob");
    strcpy( (*dpTeacher)[0].first,"Ted");

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Passing Values TO and FROM a Function

```
typedef struct human {
         char    first[32];      /* 1st field is array of char */
         char    last[32];       /* 2nd field is array of char */
         int     year;           /* 3rd field is int */
         } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| {DM} | 10000 – 10003 | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void addTeacher(person **dpT){

  dpT[0] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
  for (int i=0; i<5; i++){
        strcpy( (*dpT)[i].first,"Sam");
        strcpy( (*dpT)[i].last,"Maggs");
        (*dpT)[i].year = 2042;
  }
}
```

```
ty
in
{
   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bo
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 510 - 413 | 10000 |
| | | |
| {DM} | 10000 - 10003 | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void addTeacher(person **dpT){

  *dpT = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
  // dpT[0] = (person *) calloc ( 5, sizeof(person));
  for (int i=0; i<5; i++){
        strcpy( (*dpT)[i].first,"Sam");
        strcpy( (*dpT)[i].last,"Maggs");
        (*dpT)[i].year = 2042;
   }
}
```

```
ty

in
{
    dpTeacher = (person **)calloc(1, sizeof(person *));

    addTeacher(dpTeacher);
    printTeacher(dpTeacher);

     strcpy(dpTeacher[0][0].first,"Bo
     strcpy( (*dpTeacher)[0].first,"Te

     (*(dpTeacher+0)+2)->year = 5;

    return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 510 - 413 | 10000 |
| | | |
| {DM} | 10000 - 10003 | 10140 |
| {DM} | 10140 - 10479 | |
| | | |
| | | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void addTeacher(person **dpT){

  dpT[0] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
   for (int i=0; i<5; i++){
         strcpy( (*dpT)[i].first,"Sam");
         strcpy( (*dpT)[i].last,"Maggs");
         (*dpT)[i].year = 2042;
   }
}
```

```
ty

in
{

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bo
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 510 - 413 | 10000 |
| | | |
| dpT[0] | 10000 - 10003 | 10140 |
| {DM} | 10140 - 10479 | |
| | | |
| | | |
| | | |
| | | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void addTeacher(person **dpT){

  dpT[0] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
  for (int i=0; i<5; i++){
        strcpy( (*dpT)[i].first,"Sam");
        strcpy( (*dpT)[i].last, "Maggs");
        (*dpT)[i].year = 2042;
  }
}
```

```
ty

in
{

    dpTeacher = (person **)calloc(1, sizeof(person *));

    addTeacher(dpTeacher);
    printTeacher(dpTeacher);

     strcpy(dpTeacher[0][0].first,"Bo
     strcpy( (*dpTeacher)[0].first,"Te

     (*(dpTeacher+0)+2)->year = 5;

    return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 510 - 413 | 10000 |
| | | |
| dpt[0] | 10000 - 10003 | 10140 |
| dpt[0][0].first | 10140 - 10171 | Sam |
| dpt [0][0].last | 10172 - 10203 | Maggs |
| dpt[0][0].year | 10204 - 10207 | 2042 |
| dpt[0][1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void addTeacher(person **dpT){

  dpT[0] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
  for (int i=0; i<5; i++){
        strcpy( (*dpT)[i].first,"Sam");
        strcpy( (*dpT)[i].last, "Maggs");
        (*dpT)[i].year = 2042;
  }
}
```

```
ty

in
{

    dpTeacher = (person **)calloc(1, sizeof(person *));

    addTeacher(dpTeacher);
    printTeacher(dpTeacher);

     strcpy(dpTeacher[0][0].first,"Bo
     strcpy( (*dpTeacher)[0].first,"Te

     (*(dpTeacher+0)+2)->year = 5;

    return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 510 - 413 | 10000 |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| *dpt [0].last | 10172 - 10203 | Maggs |
| *dpt[0].year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
         char    first[32];      /* 1st field is array of char */
         char    last[32];       /* 2nd field is array of char */
         int     year;           /* 3rd field is int */
         } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| *dpt [0].last | 10172 - 10203 | Maggs |
| *dpt[0].year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void printTeacher(person **dpT){

   printf("\nThis is the contents of the dynamically allocated:\n");
   for (int i=1;i<=3;i++){
     printf("Teacher %s ",(*dpT)[i].first);
     printf("%s ",dpT[0][i].last);
     printf("%d\n\n",(*(dpT+0)+i)->year);
   }
}
```

```
ty
in
{
   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bo
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 620 - 623 | 10000 |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| *dpt [0].last | 10172 - 10203 | Maggs |
| *dpt[0].year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void printTeacher(person **dpT){

   printf("\nThis is the contents of the dynamically allocated:\n");
   for (int i=1;i<=3;i++){
     printf("Teacher %s ",(*dpT)[i].first);
     printf("%s ",dpT[0][i].last);
     printf("%d\n\n",(*(dpT+0)+i)->year);
   }
}
```

```
ty
in
{
   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bo
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 620 - 623 | 10000 |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| *dpt [0].last | 10172 - 10203 | Maggs |
| *dpt[0].year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void printTeacher(person **dpT){

   printf("\nThis is the contents of the dynamically allocated:\n");
   for (int i=1;i<=3;i++){
     printf("Teacher %s ",(*dpT)[i].first);
     printf("%s ",dpT[0][i].last);
     printf("%d\n\n",(*(dpT+0)+i)->year);
   }
}
```

```
ty
in
{
   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bo
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 620 - 623 | 10000 |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| dpt [0][1].last | 10172 - 10203 | Maggs |
| *dpt[0].year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

# POINTERS

## Passing Values TO and FROM a Function

```
void printTeacher(person **dpT){

   printf("\nThis is the contents of the dynamically allocated:\n");
   for (int i=1;i<=3;i++){
     printf("Teacher %s ",(*dpT)[i].first);
     printf("%s ",dpT[0][i].last);
     printf("%d\n\n",(*(dpT+0)+i)->year);
   }
}
```

```
ty
in
{
   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bo
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| dpT | 620 - 623 | 10000 |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| *dpt [0].last | 10172 - 10203 | Maggs |
| (*(dpT+0)+1)year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
         char    first[32];      /* 1st field is array of char */
         char    last[32];       /* 2nd field is array of char */
         int     year;           /* 3rd field is int */
         } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| *dpt | 10000 - 10003 | 10140 |
| *dpt[0].first | 10140 - 10171 | Sam |
| *dpt [0].last | 10172 - 10203 | Maggs |
| *dpt[0].year | 10204 - 10207 | 2042 |
| *dpt[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
         char    first[32];      /* 1st field is array of char */
         char    last[32];       /* 2nd field is array of char */
         int     year;           /* 3rd field is int */
         } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob
    strcpy( (*dpTeacher)[0].first,"Te

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| {DM} | 10000 - 10003 | 10140 |
| {DM} | 10140 - 10271 | ***** |
| | | |
| | | |
| | | |
| | | |
| | | |

## POINTERS

```
typedef struct human {
         char     first[32];
         char     last[32];
         int      year;
         } person;

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

   strcpy(dpTeacher[0][0].first,"Bob");
    strcpy( (*dpTeacher)[0].first,"Ted");

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| dpTeacher[0] | 10000 - 10003 | 10140 |
| dpTeacher[0][0].first | 10140 - 10171 | Bob |
| dpTeacher[0][0].last | 10172 - 10203 | Maggs |
| dpTeacher[0][0].year | 10204 - 10207 | 2042 |
| dpTeacher[0][1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

## POINTERS

```
typedef struct human {
         char     first[32];
         char     last[32];
         int      year;
         } person;

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob");
    strcpy( (*dpTeacher)[0].first,"Ted");

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

Passing Values TO and FROM a Function

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| *dpTeacher | 10000 - 10003 | 10140 |
| *dpTeacher[0].first | 10140 - 10171 | Bob |
| *dpTeacher[0].last | 10172 - 10203 | Maggs |
| *dpTeacher[0].year | 10204 - 10207 | 2042 |
| *dpTeacher[1].first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

## POINTERS

```
typedef struct human {
         char     first
         char     last[
         int      year;
         } person;

int main()
{
   person  **dpTeacher

   dpTeacher = (person

   addTeacher(dpTeache
   printTeacher(dpTeac

    strcpy(dpTeacher[0][0].first,"Bob");
    strcpy( (*dpTeacher)[0].first,"Ted");

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| *(dpTeacher+0) | 10000 - 10003 | 10140 |
| (*(dpTeacher+0)+0) ->first | 10140 - 10171 | Bob |
| (*(dpTeacher+0)+1)->last | 10172 - 10203 | Maggs |
| (*(dpTeacher+0)+2)->year | 10204 - 10207 | 2042 |
| (*(dpTeacher+1)+0)->first | 10208 - 10239 | Sam |
| {DM} | 10240 - 10271 | |
| | | |

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(1, sizeof(person *));

   addTeacher(dpTeacher);
   printTeacher(dpTeacher);

    strcpy(dpTeacher[0][0].first,"Bob");
    strcpy( (*dpTeacher)[0].first,"Ted");

    (*(dpTeacher+0)+2)->year = 5;

   return 0;
}
```

## Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(3, sizeof(person *));
   dpTeacher[0] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
   dpTeacher[1] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
   dpTeacher[2] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */

   addTeacher(dpTeacher);       /* print a part or the whole matrix */
   printTeacher(dpTeacher[1]);  /* print an entire row */

    strcpy(dpTeacher[2][3].first,"Bob");
    strcpy( (*dpTeacher+2)[0].first,"Ted");

    (*(dpTeacher+1)+2)->year = 5;

   return 0;
}
```

## POINTERS

### Passing Values TO and FROM a Function

```
typedef struct human {
        char    first[32];      /* 1st field is array of char */
        char    last[32];       /* 2nd field is array of char */
        int     year;           /* 3rd field is int */
        } person;               /* alias for human UDT ; */

int main()
{
   person  **dpTeacher;

   dpTeacher = (person **)calloc(3, sizeof(person *));
   dpTeacher[0] = (person *) calloc ( 5, sizeof(person)); /* 5 x 68 bytes */
   dpTeacher[1] = (person *) calloc (
   dpTeacher[2] = (person *) calloc (

   addTeacher(dpTeacher);          /* pr
   printTeacher(dpTeacher[1]);     /* pr

    strcpy(dpTeacher[2][3].first,"Bob
    strcpy( (*dpTeacher+2)[0].first,"

    (*(dpTeacher+1)+2)->year = 5;

   return 0;
}
```

| Label | Address | Value |
|---|---|---|
| | | |
| dpTeacher | 400 - 403 | 10000 |
| | | |
| | | |
| | | |
| {DM} | 10000 - 10011 | 10140 |
| {DM} | 10140 - 10479 | ***** |
| {DM} | 10200 - 10539 | ***** |
| {DM} | 10540 - 10879 | ***** |
| | | |
| | | |
| | | |

# Double Pointers

**END OF PART 4**