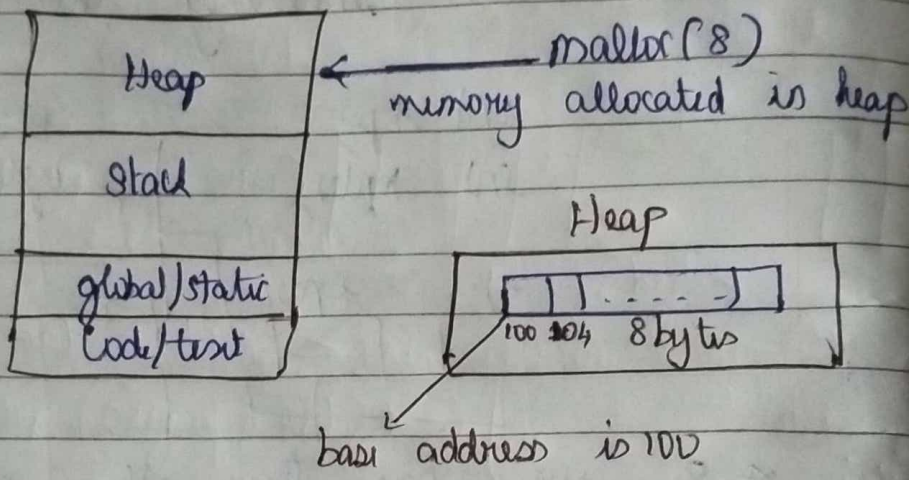
C_133 ⇒ Dynamic Memory Allocation using malloc()

* malloc() is generally used for structures and also for arrays also.

* malloc() → Memory Allocation

* If we write malloc(8); then memory of 8 bytes as one complete block is allocated in heap section and return the base address of this block.



* So obviously we need to store this base address in a pointer and hence dynamic memory allocation in heap section have pointers to point to base address of allocated memory in heap.

General Syntax:

```
void* malloc(size_t size)
```

- void* → returning address. So return type is pointer and this is void pointer
- size_t size → unsigned because memory cannot be negative.

Void Pointers :-

* Void Pointers are generic pointers which can store address of any other data types.

Normal Pointers	Void Pointers
int a ; float b;	int a, float b, char c;
int *p = &b; ✗ int *p = &a ✓	void *p1 = &a; ✓ void *p2 = &b; ✓ void *p3 = &c; ✓
printf("%d", *p); ✓ we can dereference normal pointers	printf("%d", *p2); ✗ we cannot dereference void pointers instead do typecast
	printf("%d", *(int *)p2); ✓

* Void pointers can store all type of data and hence we don't need to declare multiple pointers for multiple datatypes.

Eg: int a = 5; float b = 10.5;

① int *p;
p = &a;
✗ p = &b; → We can't store float type address in int type pointer

② void *p;
p = &a; printf("%d", *(int*)p);
p = &b; ✓
we can store any type of address in void type pointer.

* malloc() will reserve the memory section of heap of 8 bytes but right now we have no idea that which type of data is going to store and to get the output we can typecast and return the value.

* We cannot directly write malloc(8) because 8 means just a value; but we need to make understand the compiler to allocate memory in bytes so we use with sizeof operator.

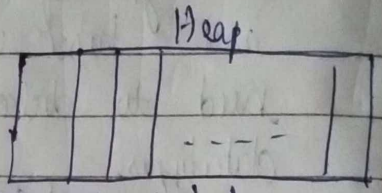
↳ Eg:- malloc(sizeof(int)*3)

↓

4 bytes × 3 ⇒ 12 bytes.

* It is a way of good practise to allocate size. Some machines take like 16/32 bit machine will have 2 bytes for int; 64 bit machine will have 4 bytes of memory for int.

void* malloc(size_t size)



int *ptr; → This pointer stores address of variable whose datatype is integer

ptr = (int*) malloc(sizeof(int)*3)

returns the base address of int type data and so we typecast the void pointer to int pointer since malloc by default returns only void pointer address

The specific typecasted pointer address is now stored in a pointer 'ptr' which going to have the typecasted pointer datatype (ie) int.

NOTE:-

* If we don't initialize these allocated memory space in heap section using malloc() then it will print any garbage values.

* If malloc() is not able to allocate memory from heap (i.e) if memory is n bytes and we give n+1 bytes to allocate memory then it leads to failure which returns null pointer; when allocates memory successfuly then it will return base address of the allocated memory block.

↳ So check in program,

```
if(ptr == NULL)
{
    printf("Error in allocating memory");
    exit(1);
}
// after doing task
free(ptr); → used to free the allocated memory in heap.
```

* Syntax:

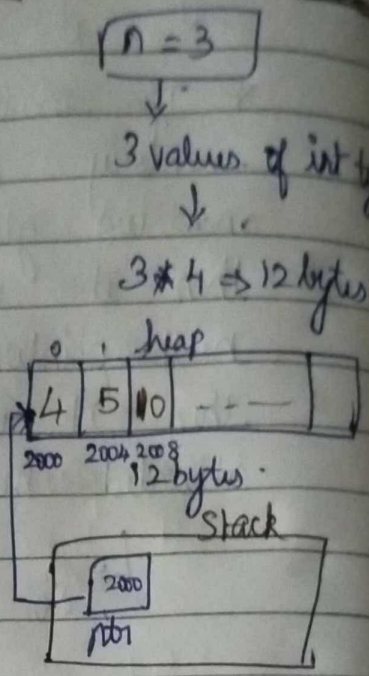
```
void* malloc(size_t size)
```

Eg:

```
int *ptr;
ptr = (int*) malloc(sizeof(int) * 3);
```

* Simple program to ask some values from user and print the value using malloc() to allocate memory in Heap and to store in that heap section.

Eg:
```
int main()
{
    int n, i, *ptr;
    printf("Enter total no. of values:\n");
    scanf("%d", &n);
    ptr = (int *) malloc(n * sizeof(int));
    printf("Enter values:");
    for(i=0; i<n; i++)
    {
        scanf("%d", ptr+i);
    }
    printf("Entered values are:");
    for(i=0; i<n; i++)
    {
        printf("%d", *(ptr+i));
    }
    free(ptr);
}
```



according to our need the memory is dynamically allocated in heap.

NOTE:

* Using a pointer only we can access the heap memory and this pointer will be allocating a memory space in stack section.

* Assignment:

Take a structure of any type like student and use malloc for the student structure.

Real Time example for SMA & DMA

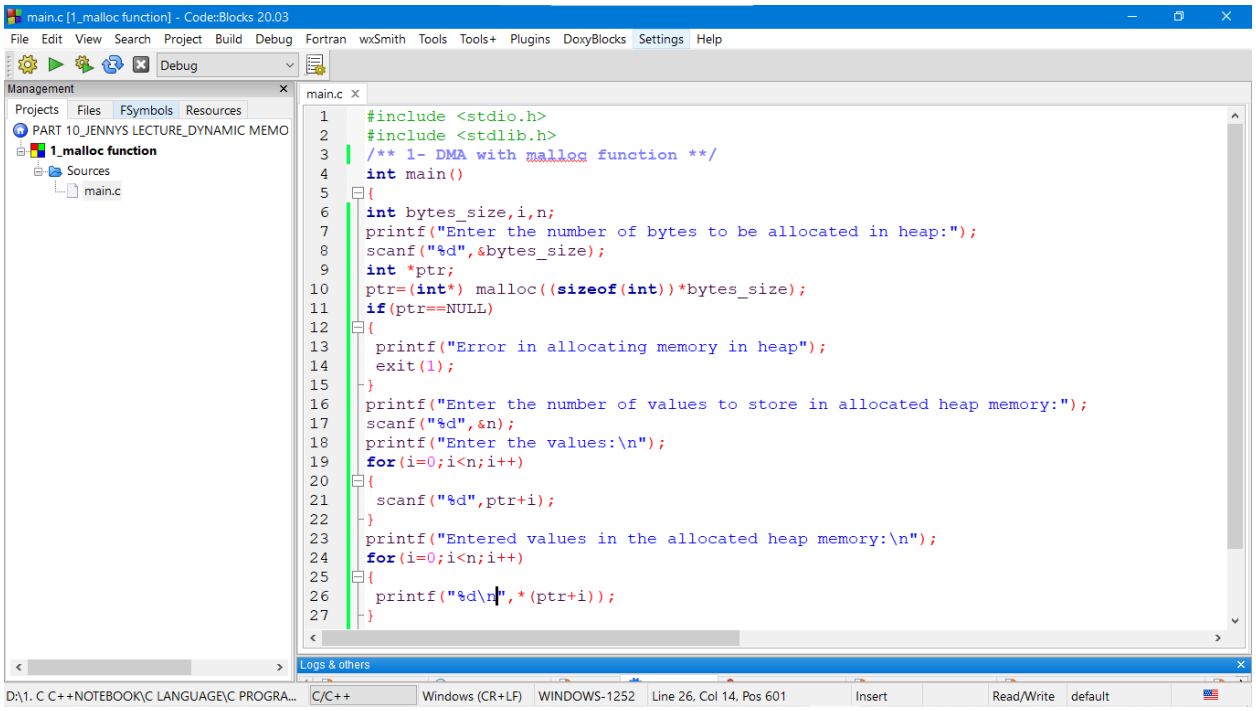
* If you are a hostler, in college, at some point of time one day you had some work so you tell your friend to bring 3 chapattis for you to eat from mess but you normally eat only two chapattis, one extra for if neassary like if chapatti nice you will eat that extra one.

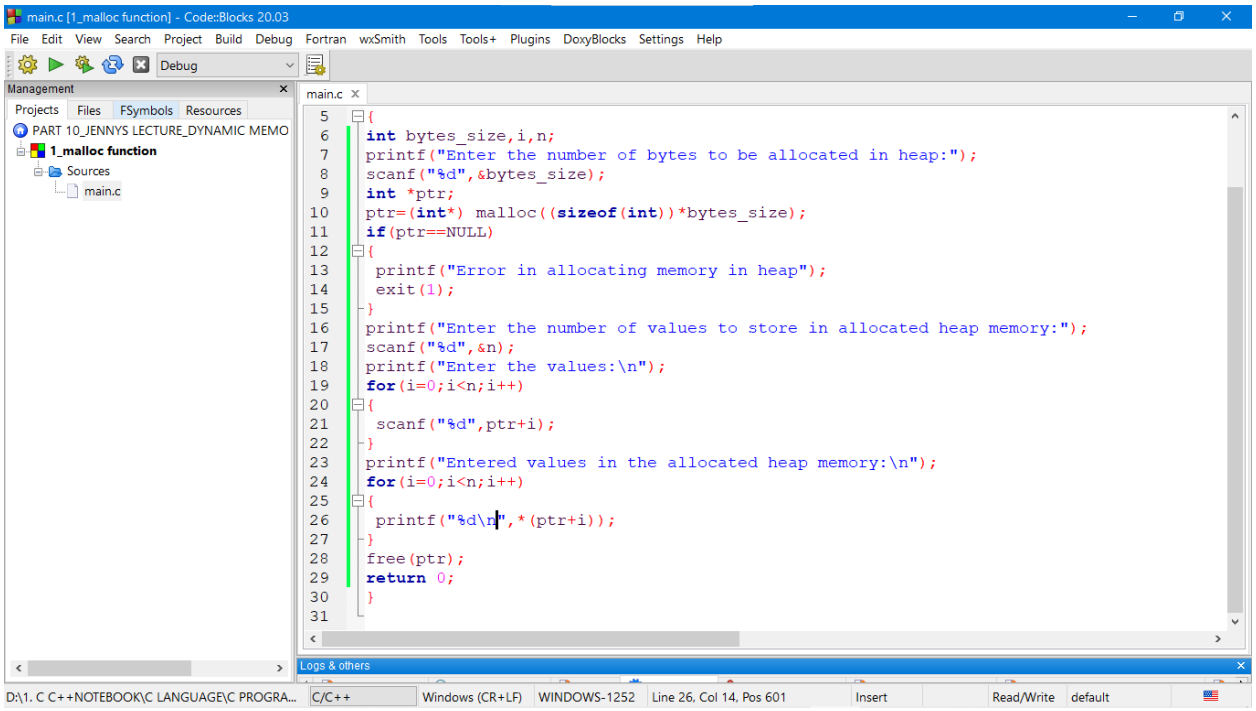
(ie) your friend is like stack section. memory is fixed.

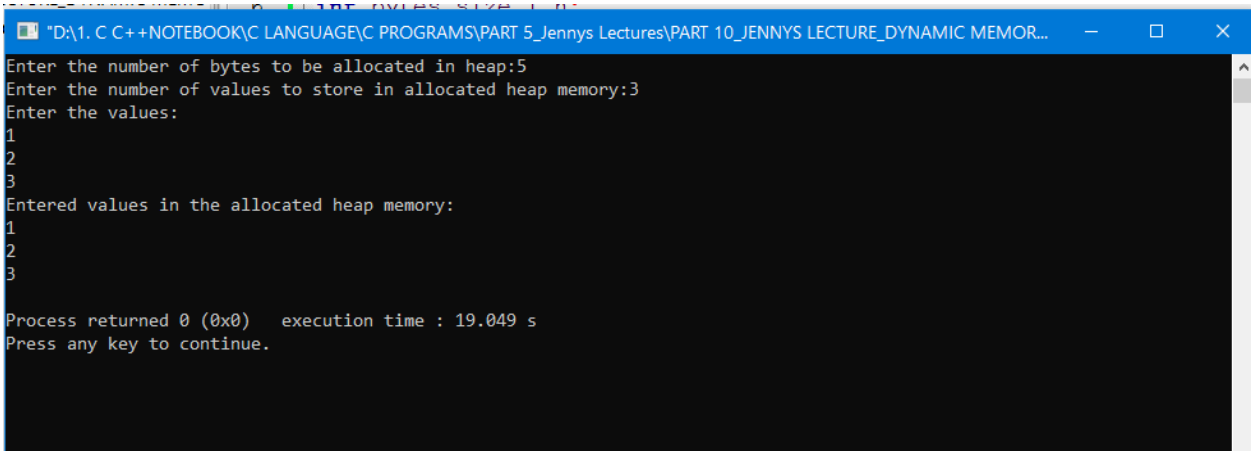
* Your friend also got 3 chapattis and you ate only two as usual since the chapattis are not that much good, one extra chapatti is now wasted. Consider this example for Static Memory Allocation

* Now instead of telling your friend to bring chapatti for you, you directly go to mess and eat the two chapatti; if you want another you eat or leave it. So consider mess as your heap section of memory and consider you are eating or consuming chapatti (memory) at the runtime and the memory is not wasted here. So this is our example for Dynamic Memory Allocation

* In SMA, your friend is like stack section and the memory allocated (3 chapattis) is fixed since you instructed him priorily (ie) at the Compile time we have given all the details for memory allocation and so it is fixed. once the Compilation is over the memory cannot be changed or alterated at run time because it is fixed is stack section

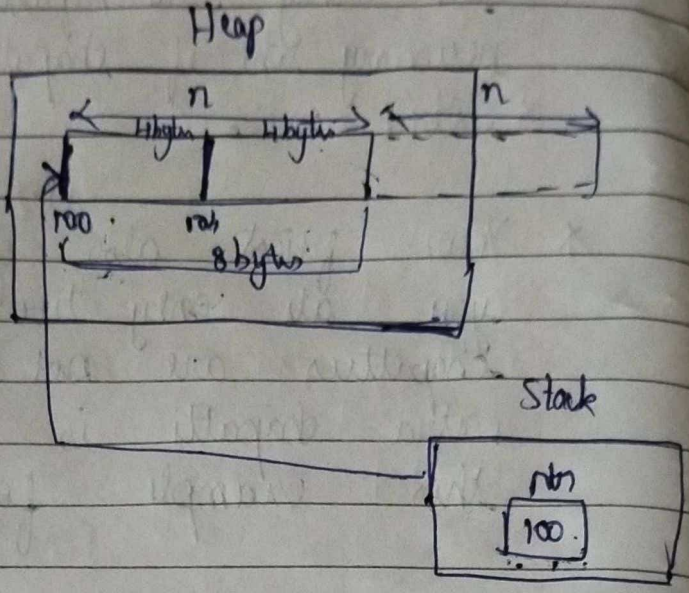

main.c [1_malloc function] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Files FSymbols Resources
PART 10_JENNYS LECTURE_DYNAMIC MEMO
1_malloc function
Sources
main.c
main.c
#include <stdio.h>
#include <stdlib.h>
/** 1- DMA with malloc function **/
int main()
{
int bytes_size,i,n;
printf("Enter the number of bytes to be allocated in heap:");
scanf("%d",&bytes_size);
int *ptr;
ptr=(int*) malloc((sizeof(int))*bytes_size);
if(ptr==NULL)
{
printf("Error in allocating memory in heap");
exit(1);
}
printf("Enter the number of values to store in allocated heap memory:");
scanf("%d",&n);
printf("Enter the values:\n");
for(i=0;i<n;i++)
{
scanf("%d",ptr+i);
}
printf("Entered values in the allocated heap memory:\n");
for(i=0;i<n;i++)
{
printf("%d\n",*(ptr+i));
}
Logs & others
D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRA...
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 26, Col 14, Pos 601
Insert
Read/Write
default


main.c [1_malloc function] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Files FSymbols Resources
PART 10_JENNYS LECTURE_DYNAMIC MEMO
1_malloc function
Sources
main.c
main.c
{
int bytes_size,i,n;
printf("Enter the number of bytes to be allocated in heap:");
scanf("%d",&bytes_size);
int *ptr;
ptr=(int*) malloc((sizeof(int))*bytes_size);
if(ptr==NULL)
{
printf("Error in allocating memory in heap");
exit(1);
}
printf("Enter the number of values to store in allocated heap memory:");
scanf("%d",&n);
printf("Enter the values:\n");
for(i=0;i<n;i++)
{
scanf("%d",ptr+i);
}
printf("Entered values in the allocated heap memory:\n");
for(i=0;i<n;i++)
{
printf("%d\n",*(ptr+i));
}
free(ptr);
return 0;
}
Logs & others
D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRA...
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 26, Col 14, Pos 601
Insert
Read/Write
default


"D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 10_JENNYS LECTURE_DYNAMIC MEMOR...
Enter the number of bytes to be allocated in heap:5
Enter the number of values to store in allocated heap memory:3
Enter the values:
1
2
3
Entered values in the allocated heap memory:
1
2
3
Process returned 0 (0x0)   execution time : 19.049 s
Press any key to continue.

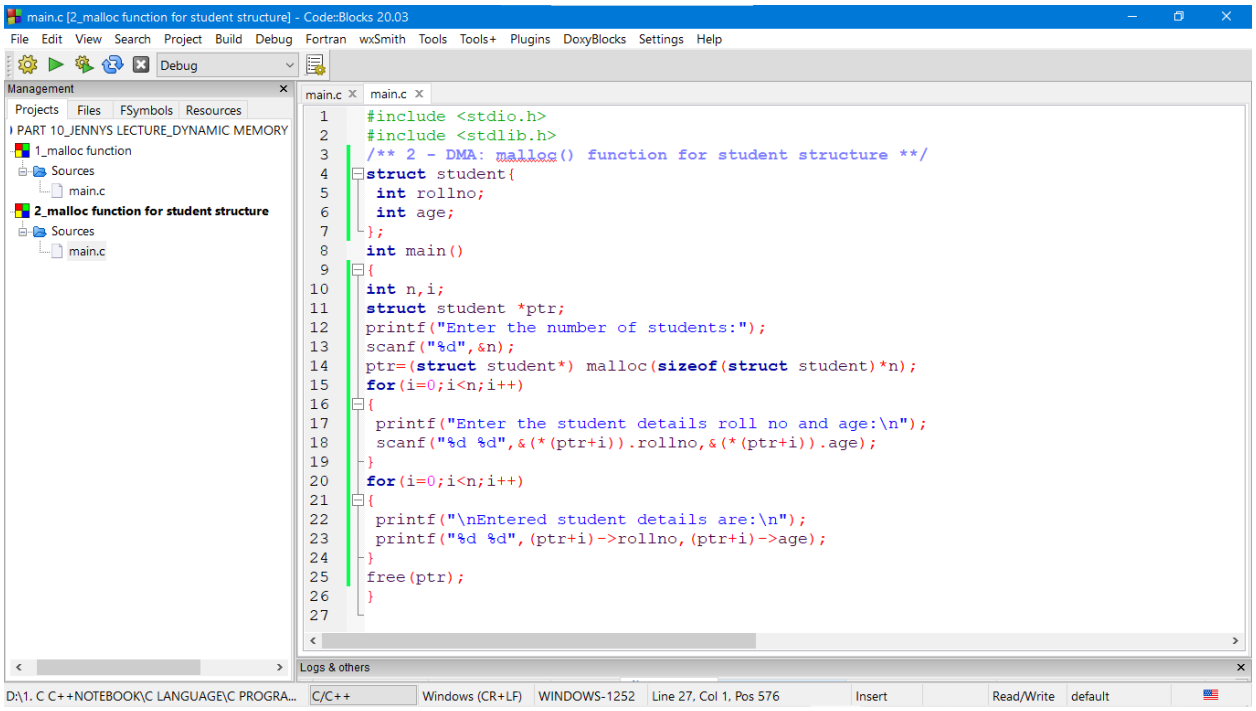
Assignment:

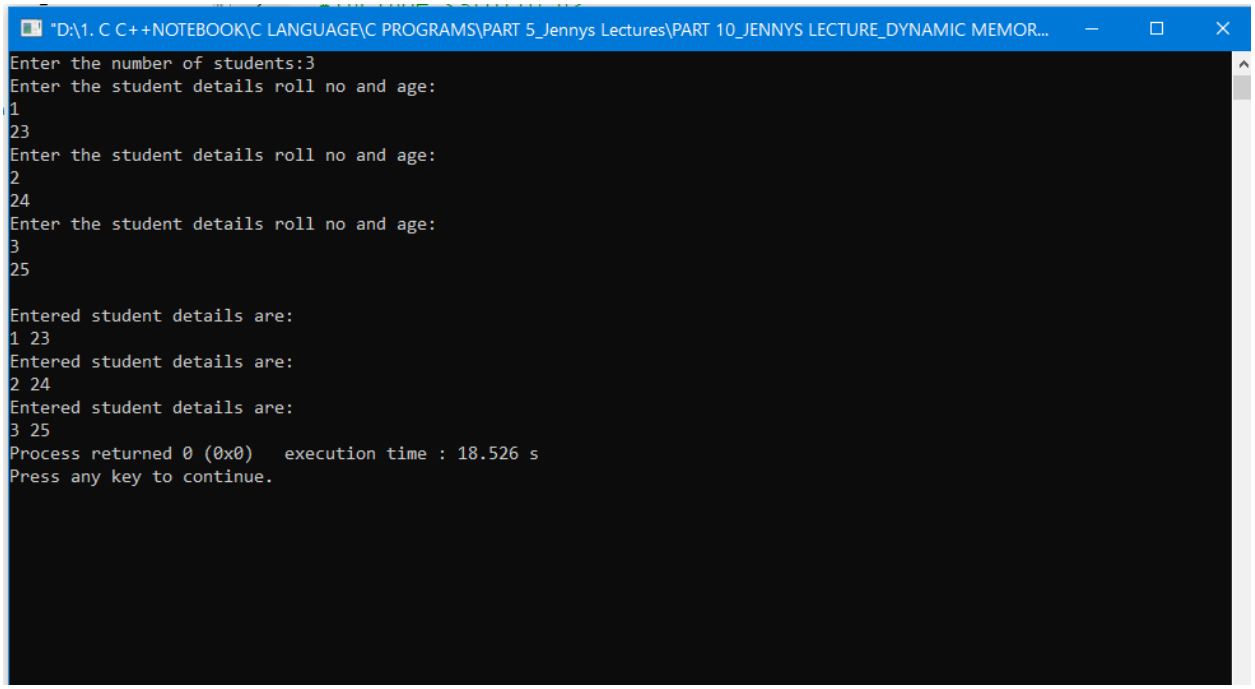
DMA → Memory allocation malloc() for student structure




```
Struct student
{
  int rollno;
  int age;
};

int main()
{
  int n, i;
  struct student *ptr;
  printf("Enter the number of students:");
  scanf("%d", &n);
  ptr = (struct student *) malloc (sizeof(struct student)
                                   * n);
  for(i=0; i<n; i++)
  {
    printf("Enter the student details,
                rollno and age:\n");
    scanf("%d %d", &(*(ptr+i)).rollno,
                   &(*(ptr+i)).age);
  }
  for(i=0; i<n; i++)
  {
    printf("\n Entered student details are:\n");
    printf("%d %d", (ptr+i)->rollno,
                    (ptr+i)->age);
  }
  free(ptr);
}
```


main.c [2_malloc function for student structure] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Files FSymbols Resources
PART 10_JENNYS LECTURE_DYNAMIC MEMORY
1_malloc function
Sources
main.c
2_malloc function for student structure
Sources
main.c
main.c
main.c
#include <stdio.h>
#include <stdlib.h>
/** 2 - DMA: malloc() function for student structure **/
struct student{
int rollno;
int age;
};
int main()
{
int n,i;
struct student *ptr;
printf("Enter the number of students:");
scanf("%d",&n);
ptr=(struct student*) malloc(sizeof(struct student)*n);
for(i=0;i<n;i++)
{
printf("Enter the student details roll no and age:\n");
scanf("%d %d",&(*(ptr+i)).rollno,&(*(ptr+i)).age);
}
for(i=0;i<n;i++)
{
printf("\nEntered student details are:\n");
printf("%d %d",(ptr+i)->rollno,(ptr+i)->age);
}
free(ptr);
}
Logs & others
D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRA...
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 27, Col 1, Pos 576
Insert
Read/Write
default


"D:\1. C C++NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 10_JENNYS LECTURE_DYNAMIC MEMOR...
Enter the number of students:3
Enter the student details roll no and age:
1
23
Enter the student details roll no and age:
2
24
Enter the student details roll no and age:
3
25
Entered student details are:
1 23
Entered student details are:
2 24
Entered student details are:
3 25
Process returned 0 (0x0) execution time : 18.526 s
Press any key to continue.