CS101: Introduction to Programming

Venkatesh Vinayakarao
Department of Computer Science and Engineering
IIIT Sri City, Chittoor.
venkatesh.v@iiits.in

Online Judges

- Please try https://uva.onlinejudge.org/index.php
- Solve problem **136 - Ugly Numbers**
- No need to submit anything. Not graded.

Bonus Task – 4% Marks

- Applies only if the total (after including the bonus) < 90%.
- Only top-6 submissions by quality will get the bonus marks.
- Instructor judgment is final.
- Deadlines are strict.

Bonus Task 1

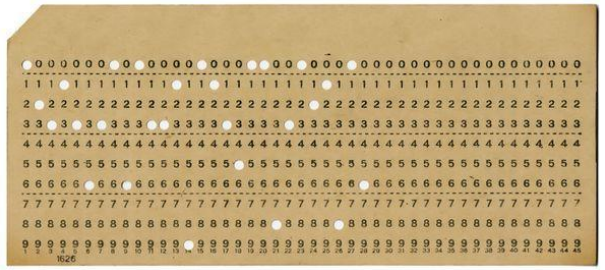
- Make a 5 to 10 slide presentation on Compilers
- Your 15 minute presentation should cover all (but not limited to) the following topics:
  - Variety of C compilers
  - What does a C compiler do?
  - Give the history of C compilers.
  - An example of code which gives different output in different compilers
  - Which compiler should I use and why?
- Deadline: 18th Sep 2018.

Agenda

- Pointers
  - Introduction

History of Computers

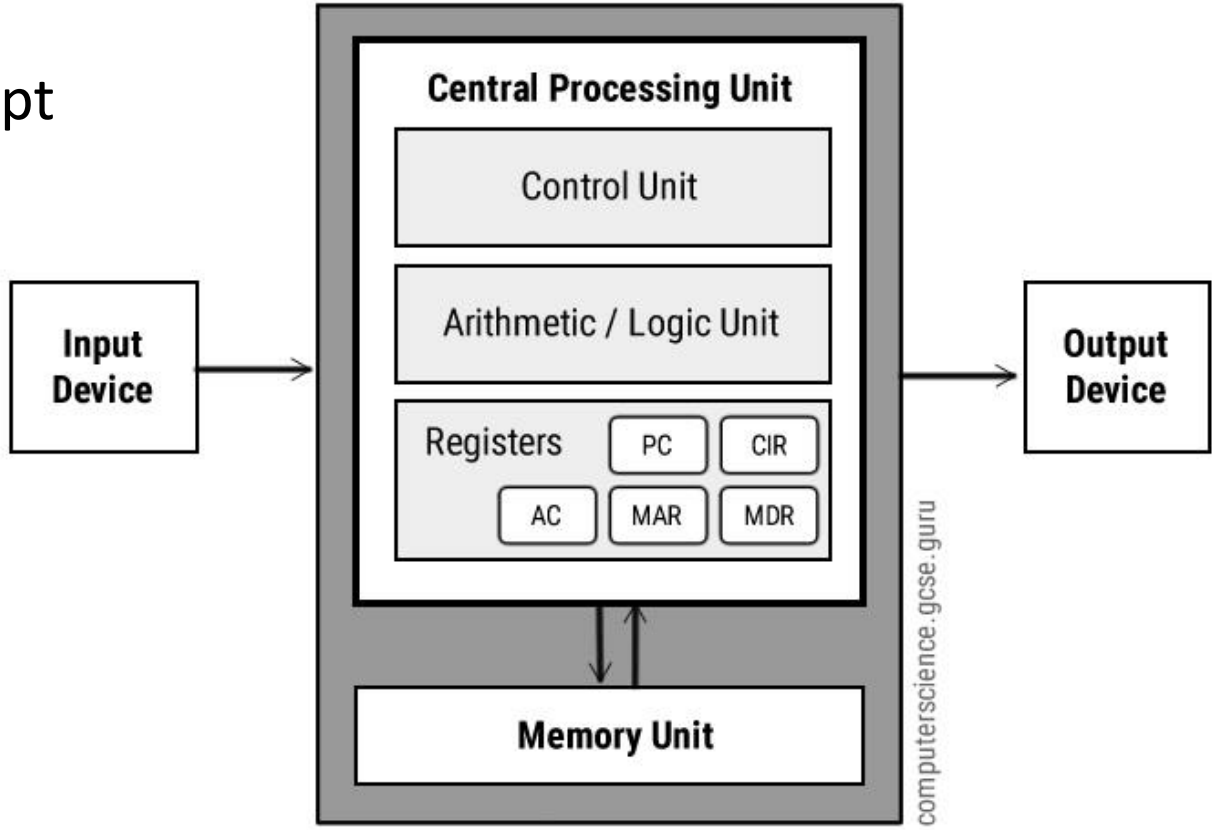
- Earlier computers could not store data or programs.
- Punch Cards were used to store data and programs as early as 1725.
  - In 1890, Herman Hollerith developed machines to read and write punch cards.
  - He later formed IBM.
  - Thus, punch cards are also known as Hollerith Cards or IBM Cards.



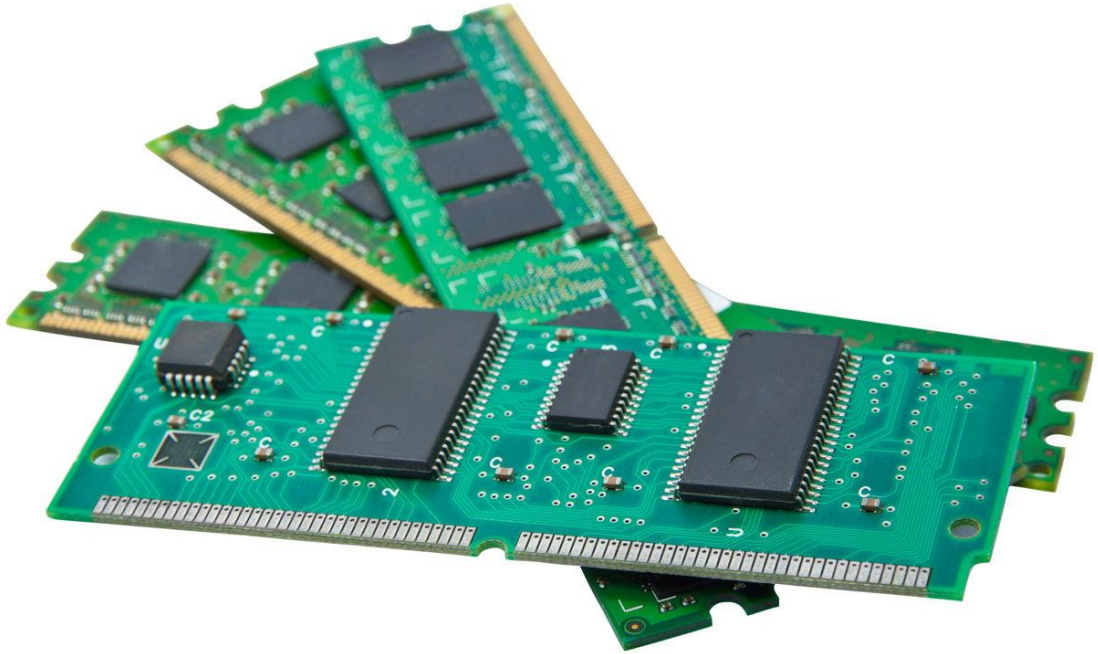
ComputerHope.com

How does it work?

- Von Neumann Architecture
  - 1940 – Stored Program Concept

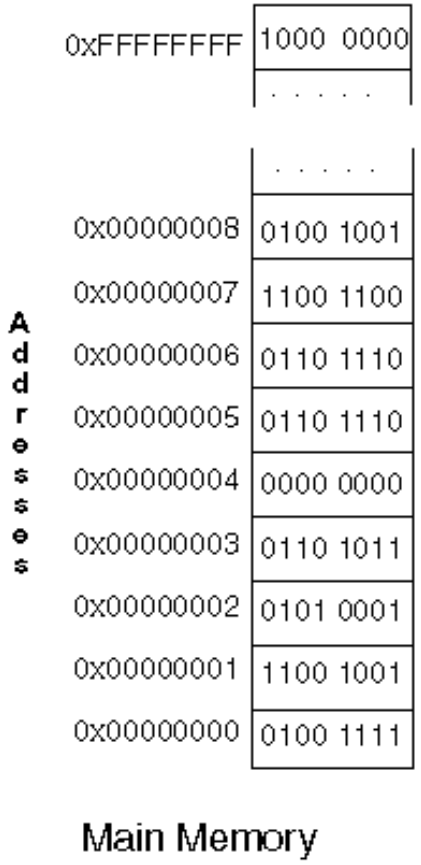


Computer Memory



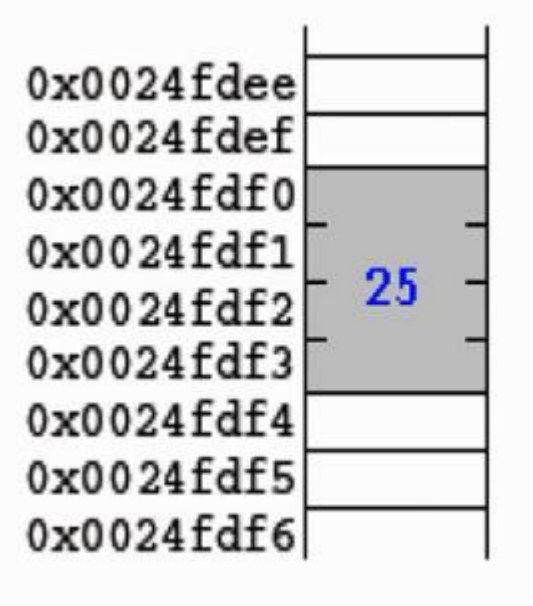
Memory and Addresses

- Computer with 1 GB RAM has an array of 1024 * 1024 * 1024 Bytes.



Variable Declaration

- What happens when int x = 25; is run?
  - Depending on the data type (int in this case), some amount of memory is reserved.
  - The value 25 is written to this memory location.



3 is printed…

```
1  int main() {
2      int i = 3;
3      printf("%d", i);
4  }
```

Where is i stored?

```c
int main() {

    int i = 4;
    printf("%u\n", &i);
    printf("%d\n", *(&i));
}
```

Memory address cannot be negative. %u refers to unsigned int.

&i refers to address of i.

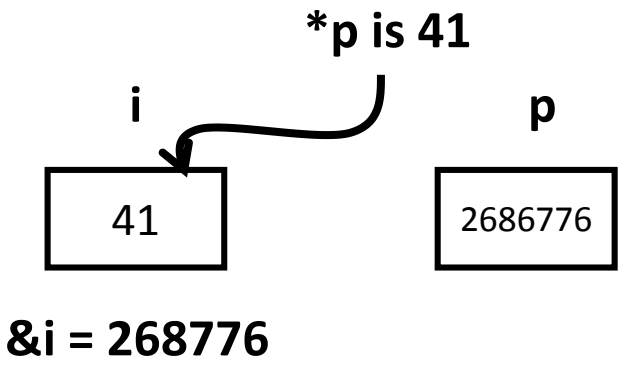
*(&i) refers to the value at the address of i.

```
2686780
4

Process returned 0 (0x0)
Press any key to continue.
```

A Pointer Variable

- int *p is a declaration of a pointer variable p.
- We say that p points to some location.

```c
int main() {

    int i = 41;

    int *p;
    p = &i;

    printf("%u\n", p);
    printf("%d\n", *p);
}
```




```
2686776
41

Process returned 0 (0x0)
Press any key to continue.
```

Pointers

- What is the value of i?
- What is the address of i?

i

41

&i = 268776

A memory snapshot while running your program

Quiz

- What is the value of i?
- What is the address of i?

i

42

&i = 268776

A memory snapshot while running your program

Quiz

- What is the memory snapshot of int i = 43; ?

i

43

&i = 268776

A memory snapshot while running your program

Quiz

- What is the memory snapshot of int i = 44; ?

i

44

&i = 268776

A memory snapshot while running your program

Quiz

- What is the memory snapshot of the following code?

```
int i = 44;
int j = 45;
```

Memory and Variables

What is the memory snapshot of the following code?

```c
int main() {

    int i = 44;
    int j = 45;

    printf("%u\n", &i);
    printf("%u\n", &j);
}
```

i	j
44	45
&i = 2686780	**&j = 2686776**
A memory snapshot while running your program	A memory snapshot while running your program

```
2686780
2686776

Process returned 0 (0x0)
Press any key to continue.
```

Quiz

- What is the memory snapshot of the following code?

```
char i = ‘a’;
char j = ‘b’;
```

Quiz

- What is the memory snapshot of the following code?

char i = ‘a’;

char j = ‘b’;

```
int main() {

    char i = 'a';
    char j = 'b';

    printf("%u\n", &i);
    printf("%u\n", &j);
}
```

char occupies only 1 byte!

i

a

&i = 2686783

A memory snapshot while running your program

j

b

&j = 2686782

A memory snapshot while running your program

```
2686783
2686782

Process returned 0 (0x0)
Press any key to continue.
```

Quiz

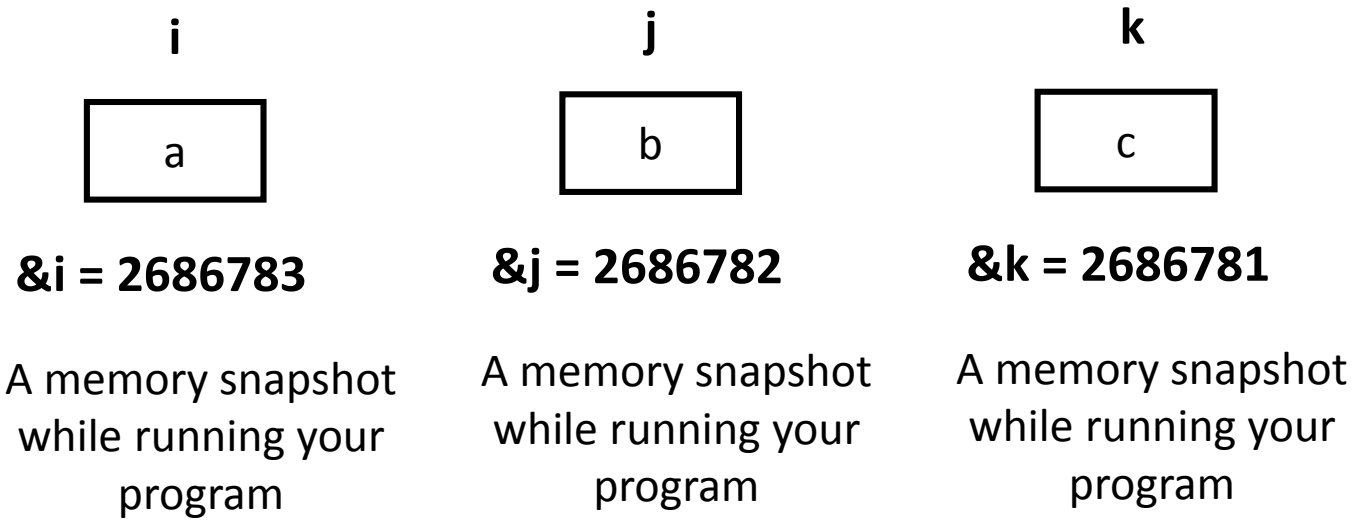
- What is the memory snapshot of the following code?

```
char i = ‘a’;
char j = ‘b’;
char k = ‘c’;
```

Quiz

- What is the memory snapshot of the following code?

```
char i = 'a';
char j = 'b';
char k = 'c';
```



Quiz

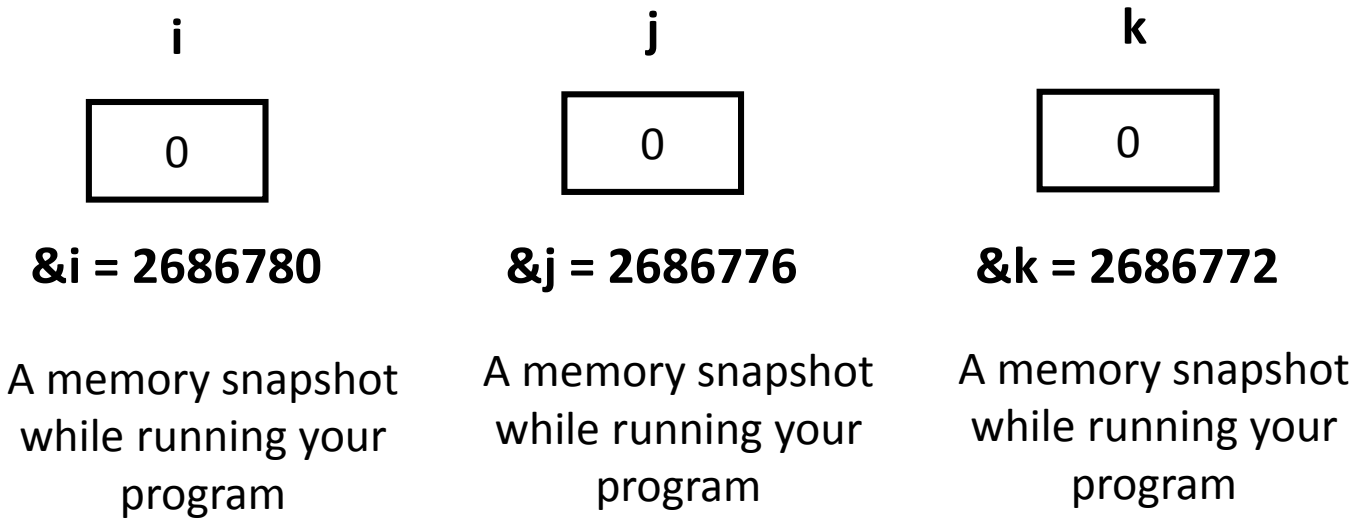
- What is the memory snapshot of the following code?

```
int i = 0;
int j = 0;
int k = 0;
```

Quiz

- What is the memory snapshot of the following code?

```
int i = 0;
int j = 0;
int k = 0;
```



Quiz

- What is the memory snapshot of the following code?

```
int i = 0;
char j = ‘a’;
int k = 0;
```

Quiz

- What is the memory snapshot of the following code?

```
int i = 0;
char j = 'a';
char k = 'a';
```

i	j	k
0	a	a
&i = 2686780	**&j = 2686779**	**&k = 2686778**
A memory snapshot while running your program	A memory snapshot while running your program	A memory snapshot while running your program

```
2686780
2686779
2686778

Process returned 0 (0x0)
Press any key to continue.
```

The right way to print addresses

```c
int main() {

    int i = 0;
    char j = 'a';
    char k = 'b';



    printf("%p\n", &i);
    printf("%p\n", &j);
    printf("%p\n", &k);
}
```

```
0028FF3C
0028FF3B
0028FF3A

Process returned 0 (0x0)
Press any key to continue.
```

Hexadecimal Number System

- 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

0_{hex}	=	0_{dec}
1_{hex}	=	1_{dec}
2_{hex}	=	2_{dec}
3_{hex}	=	3_{dec}
4_{hex}	=	4_{dec}
5_{hex}	=	5_{dec}
6_{hex}	=	6_{dec}
7_{hex}	=	7_{dec}
8_{hex}	=	8_{dec}
9_{hex}	=	9_{dec}
A_{hex}	=	10_{dec}
B_{hex}	=	11_{dec}
C_{hex}	=	12_{dec}
D_{hex}	=	13_{dec}
E_{hex}	=	14_{dec}
F_{hex}	=	15_{dec}

A Variety of Applications

- http://www.example.com/name%20with%20spaces

What is %20?

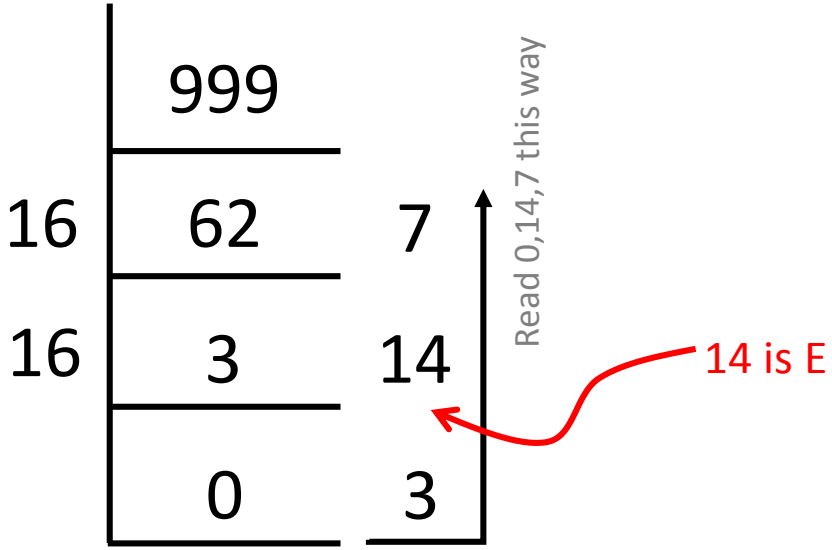
%20 in hex = 32 in decimal = space in ASCII

Memory Addresses

- Usually memory is addressed in Hexadecimal units for convention and convenience
  - bits, bytes, KB, MB, GB are all powers of 2.
  - A number system such as hexadecimal makes it convenient.
  - Binary is too long.
  - DEC used octal. IBM used Hex. We follow it ever since then.

Base Conversion

Convert 999 from decimal to hex.



$999_{10} = 0x3E7_{16}$

Decimal	Hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Quiz: Covert Decimal to Hex

- Sometimes, beauty emerges from unexpected corners!

Decimal	Hex
256	
512	
1024	
2048	
4096	

Quiz

Decimal	Hex
256	0x100
512	0x200
1024	0x400
2048	0x800
4096	0x1000

Have you ever wondered why RAM sizes come in these sizes?

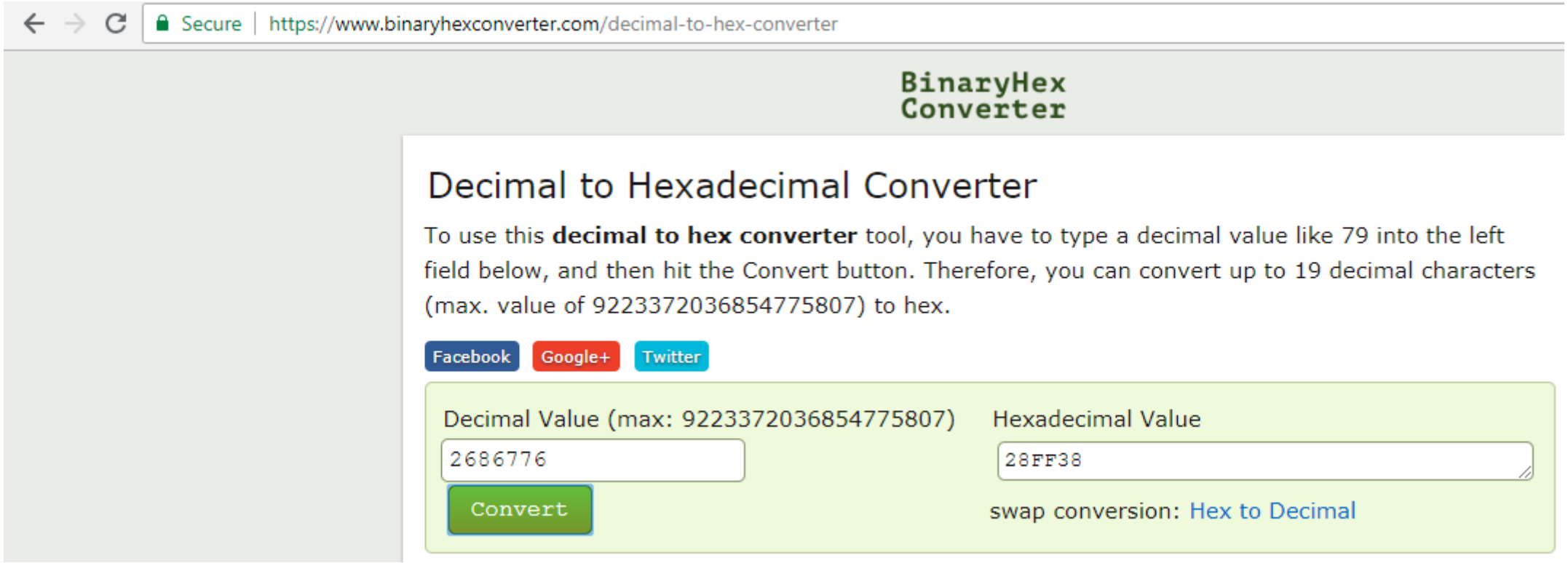
Convert Decimal to Hex in C

```c
int main() {

    int x = 25;

    int *p;
    p = &x;

    printf("The hex value of %u is %x\n", p, p);
}
```

C:\ccode\pointer18-hex.exe

```
The hex value of 2686776 is 28ff38

Process returned 0 (0x0)   execution time
Press any key to continue.
```

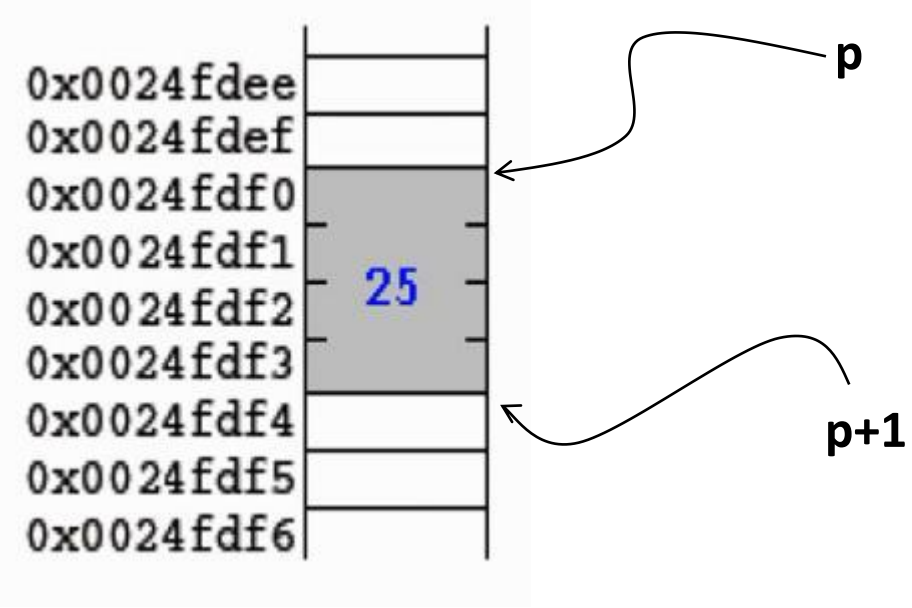
Check it Online!



Pointer Arithmetic

Memory

- p points to x.
  - int x = 25;
  - int *p;
  - p = &x;
- What is p+1 pointing to?
  - p+1 points to x + sizeof(x)



Pointer Arithmetic

```c
int main() {

    int x = 25;
    int *p;
    p = &x;

    printf("%u\n", p);
    printf("%u\n", p+1);
}
```

C:\ccode\pointer15.exe

```
2686776
2686780

Process returned 0 (0x0)
Press any key to continue.
```

Is y same as *(p+1)?

```c
int main() {

    int x = 25;
    int y = 10;

    int *p0;
    p0 = &x;

    int *p1;
    p1 = &y;

    printf("The value at %u is %d\n", p0, *p0);
    printf("The value at %u is %d\n", p1, *p1);
}
```

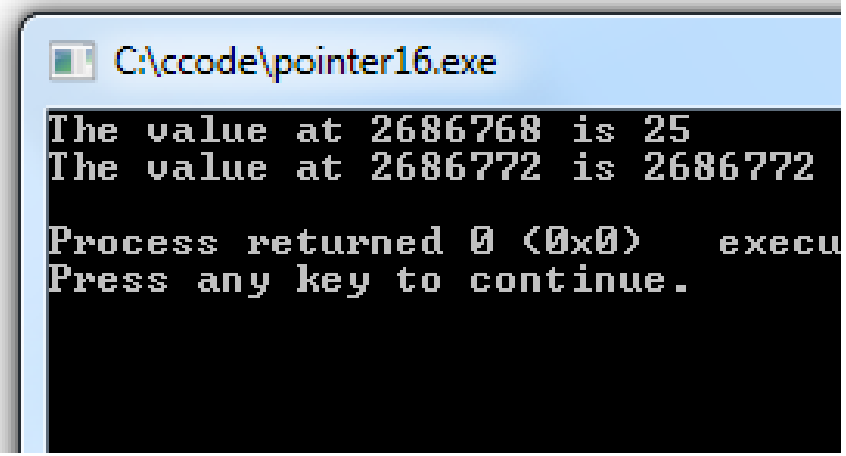
C:\ccode\pointer16.exe

```
The value at 2686772 is 25
The value at 2686768 is 10

Process returned 0 (0x0)
Press any key to continue.
```

Cannot do this!

```c
int main() {

    int x = 25;
    int y = 10;

    int *p0;
    p0 = &x;

    int *p1;
    p1 = p0 + 1;

    printf("The value at %u is %d\n", p0, *p0);
    printf("The value at %u is %d\n", p1, *p1);
}
```

Next variable need not be stored in the next memory location.



Quiz: What is the Output?

```c
int main() {

    char c0 = 'a';
    char c1 = 'b';

    char *p0;
    p0 = &c0;

    char *p1;
    p1 = &c1;

    printf("The value at %u is %c\n", p0, *p0);
    printf("The value at %u is %c\n", p1, *p1);
}
```

C:\ccode\pointer17.exe

```
The value at 2686775 is a
The value at 2686774 is b

Process returned 0 (0x0)
Press any key to continue.
```

Quiz: What is the Wrong?

```c
int main() {

    char c0 = 'a';
    char c1 = 'b';

    int *p0;
    p0 = &c0;

    int *p1;
    p1 = &c1;

    printf("The value at %u is %d\n", p0, *p0);
    printf("The value at %u is %d\n", p1, *p1);
}
```

C:\ccode\pointer17.exe

```
The value at 2686775 is 687814241
The value at 2686774 is -13213342

Process returned 0 (0x0)   executi
Press any key to continue.
```

Functions and Pointers

Functions - Revision

```
1  int main() {
2      int x = 20;
3      change(x);
4      printf("%d", x);
5  }
6
7  void change(int x) {
8      x = 10;
9  }
```

Functions – Call by value

```
1  int main() {
2      int x = 20;
3      change(x);
4      printf("%d", x);
5  }
6
7  void change(int x) {
8      x = 10;
9  }
```

Nothing changes.

x is printed as 20.

Quiz: Functions

```c
#include <stdio.h>

int main () {
    int i = 10;
    check(i);
    printf("%d", i);
}

int check(int i) {
    return i + 1;
}
```

Quiz: Functions

```c
#include <stdio.h>

int main () {
    int i = 10;
    check(i);
    printf("%d", i);
}

int check(int i) {
    return i + 1;
}
```

Prints 10

Call by Value

- Be Careful!

```c
#include <stdio.h>

int main () {
    int i = 10;
    printf("%d", check(i));
}

int check(int i) {
    return i + 1;
}
```

Prints 11

Call by Reference

```
1  int main() {
2      int x = 20;
3      change(&x);
4      printf("%d", x);
5  }
6
7  void change(int *x) {
8      *x = 10;
9  }
```

Call by Reference

```c
int main() {
    int x = 20;
    change(&x);
    printf("%d", x);
}

void change(int *x) {
    *x = 10;
}
```

x

20

&x = 2686780

A memory snapshot while running your program

change(&x) copies the address i.e., 2686780.

***x = 10 changes the value at address &x.**

main() prints 10.

Quiz

```
2   int sum(int a, int b)
3   {
4        int c=a+b;
5        return c;
6   }
7
8   int main()
9   {
10      int var1 =10;
11      int var2 = 20;
12      int var3 = sum(var1, var2);
13      printf("%d", var3);
14      return 0;
15  }
```

Prints 30

Quiz

```
2   int sum(int a, int b)
3   {
4        int c=a+b;
5        return c;
6   }
7
8   int main()
9   {
10      int var1 =20;
11      int var2 = 30;
12      int var3 = sum(var1, var2);
13      printf("%d", var3);
14      return 0;
15  }
```

Prints 50

Quiz

```
2   int sum(int a, int b)
3   {
4        int c=a+b;
5        return c;
6   }
7
8   int main()
9   {
10      int var1 =20;
11      int var2 = 30;
12      int var3 = sum(var1, var2);
13      printf("%d", var3);
14      return 0;
15  }
```

Prints 50

Quiz

```
2   int sum(int a, int b)
3   {
4        int c=a+b;
5        return 0;
6   }
7
8   int main()
9   {
10       int var1 =20;
11       int var2 = 30;
12       int var3 = sum(var1, var2);
13       printf("%d", var3);
14       return 0;
15  }
```

Prints 0

Quiz

- Oh No!!! What happened?

```
2   int sum(int a, int b)
3   {
4        int c=a+b;
5   }
6
7   int main()
8   {
9       int var1 =20;
10      int var2 = 30;
11      int var3 = sum(var1, var2);
12      printf("%d", var3);
13      return 0;
14  }
```

The confusing interplay of registers!

Quiz

```
2   int sum(int a, int b)
3   {
4        a = 1;
5        b = 2;
6   }
7
8   int main()
9   {
10      int x =20;
11      int y = 30;
12
13      sum(x, y);
14
15      printf("%d %d", x, y);
16
17      return 0;
18  }
```

Prints 20 30

Quiz

```
2   int sum(int x, int y)
3   {
4        x = 1;
5        y = 2;
6   }
7
8   int main()
9   {
10      int x =20;
11      int y = 30;
12
13      sum(x, y);
14
15      printf("%d %d", x, y);
16
17      return 0;
18  }
```

Prints 20 30

Quiz

```
2   int sum(int x, int y)
3   {
4        x = 1;
5        y = 2;
6   }
7
8   int main()
9   {
10      int x =20;
11      int y = 30;
12
13      sum(x, y);
14
15      printf("%d %d", x, y);
16
17      return 0;
18  }
```

Prints 20 30

Pointers

```
2   int sum(int *x, int *y)
3   {
4        *x = 1;
5        *y = 2;
6   }
7
8   int main()
9   {
10      int x =20;
11      int y = 30;
12
13      sum(&x, &y);
14
15      printf("%d %d", x, y);
16
17      return 0;
18  }
```

Arrays and Pointer Arithmetic

Arrays

```
1  int main () {
2
3      int  var[] = {10, 100, 200};
4      for(int i=0; i<=2; i++) {
5          printf("%d ", var[i]);
6      }
7  }
```

Arrays with Pointers

```
1   int main () {
2
3       int  var[] = {10, 100, 200};
4       int *ptr;
5       ptr = var;
6       for(int i=0; i<=2; i++) {
7           printf("%d ", *ptr);
8           ptr++;
9       }
10  }
```