Space Complexity

Datatype	C (in Bytes)	Java	Python
Int	2 (or) 4	4	4
Char	1	2	2
Float	4	4	4
Double	8	8	Doesn't exist
Long	4 (or) 8	8	Doesn't exist
Short	2	2	2
Byte	Doesn't exist	1	Doesn't exist
Bool	Doesn't exist	1 bit	1 bit

Signed Char –

1 byte = 8 bits

1 bit used for sign, remaining 7 are left

So, it ranges from -2^7 to (2^7)-1 (-128 to 127)

Unsigned Char-

1 byte = 8 bits

So, it ranges from 0 to 2^8 (0 to 255)

Signed Integer –

Integer is 4 bytes

4 bytes = 32 bits

1 bit used for sign, remaining 31 are left

So, it ranges from -2^31 to (2^31)-1 (-2147483648 to 2147483647)

Unsigned Integer –

Integer is 4 bytes

4 bytes = 32 bits

So, it ranges from 0 to 2^32 (0 to 4,294,967,295)

Space Complexity in Structures

Space Complexity in Unions and Structures depends on Cycle Time. Cycle time is equal to highest of the datatype.

Struct apple
{

Int a; // 4

Float b; // 4

Char c; // 1
}

Size of this struct is 12

Struct apple
{

Int a; // 4

Double b; // 8

Char c; // 1
}

Size of this struct is 24

Struct apple
{

Int a;

Float b;

Char c[100]; // 100 bytes = 25 cycles (no remainder so perfectly covered)
}

Size of this struct is 108

Struct apple
{

Int a;// 4 + 4 since cycle time is 8.

Double b; // 8

Char c[777]; // 97.125 (take 7 bytes more since remainder is 1) so 777+7
}

Size of this struct is 800

Struct apple
{

Int a; 4 + 4

Double b; 8

Char c[1999]; // remainder is 7 add 1 extra so 2000
}

Size of this struct is 2016

Struct apple
{

Int a[100]; // 400

Double b; // 8

Char c[1999];// 2000
}

Size of this struct is 2408

Struct apple
{

Int a[103]; // 412+4

Double b; // 8

Char c[1999];// 2000
}

Size of this struct is 2424

Struct apple
{

Int a[103]; // 412+4

Double b[3000]; // 3000*8=24000

Char c[1999];// 2000
}

Size of this struct = 26,416

Space Complexity in Unions

Union apple
{

Int a; // 4

Float b; // 4

Char c; // 1
}

Size of this union is 4

Union apple
{

Int a; // 4

Double b; // 8

Char c; // 1
}

Size of this union is 8

union apple
{

Int a;

Float b;

Char c[100]; // 100 bytes = 25 cycles (no remainder so perfectly covered)
}

Size of this union is 100

union apple
{

Int a;// 4 + 4 since cycle time is 8.

Double b; // 8

Char c[777]; // 97.125 (take 7 bytes more since remainder is 1) so 777+7
}

Size of this union is 784

Union apple
{

Int a; 4 + 4

Double b; 8

Char c[1999]; // remainder is 7 add 1 extra so 2000
}

Size of this union is 2000

Union apple
{

Int a[100]; // 400

Double b; // 8

Char c[1999];// 2000
}

Size of this union is 2000

Union apple
{

Int a[103]; // 412+4

Double b; // 8

Char c[1999];// 2000
}

Size of this union is 2000

Union apple
{

Int a[103]; // 412+4

Double b[3000]; // 3000*8=24000

Char c[1999];// 2000
}

Size of this union = 24000

HW for today

1. Real time examples for OOPC
2. Magical Prime and Neon Number using Inheritance
3. How can be calculate Space Complexity in Union?

Object Oriented Programming Concepts

Compile Time are stored in Stack(ordered) and Runtime is stored in Heap.

Temp data in Heap and Permanent data is in stored in Stack.

All Objects are stored in Heap Memory (unordered).

Inheritance

Single Level , Multi Level , Multiple , Hierarchical (check programs attached in repo)

Real Time Examples for OOPC Concepts:

Encapsulation : Used to hide the implementation of code (like ATM , Supermarket , Credit Card machine)

Polymorphism : Duck Typing, Hierarchy based coding

Inheritance : reusing of same classes , code reusability from parent classes

Abstraction : Simplify coding stuff